(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
22 March 2001 (22.03.2001)

PCT

(10) International Publication Number
WO 01/20501 A1

(51) International Patent Classification[7]: G06F 17/30

(21) International Application Number: PCT/US00/07416

(22) International Filing Date: 20 March 2000 (20.03.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/394,218 13 September 1999 (13.09.1999) US

(71) Applicant: VITESSE SEMICONDUCTOR CORPORATION [US/US]; 741 Calle Plano, Camarillo, CA 93012 (US).

(72) Inventors: TRIPATHI, Devendra, K.; 1825 Canal Way, San Jose, CA 95132 (US). DEB, Alak, K.; 3230 Vintage Crest Drive, San Jose, CA 95148 (US).

(74) Agent: OLYNICK, Mary, R.; Beyer Weaver & Thomas, LLP, P.O. Box 130, Mountain View, CA 94042-0130 (US).

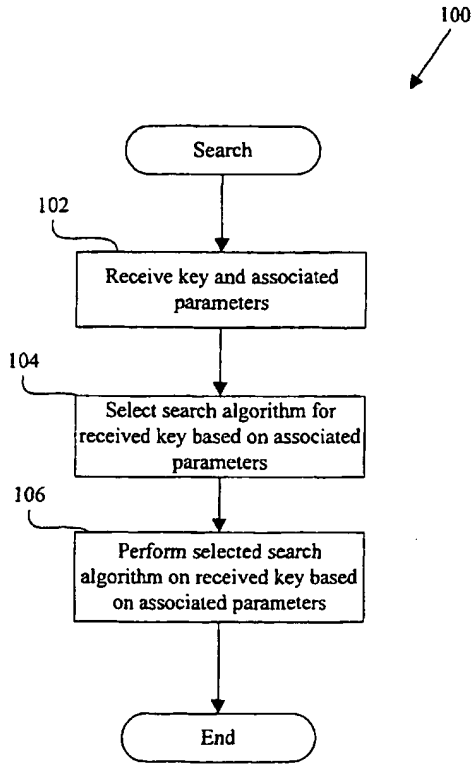
(81) Designated States *(national)*: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *With international search report.*

[Continued on next page]

(54) Title: SEARCH ALGORITHM METHODS AND APPARATUS



(57) Abstract: Disclosed is an apparatus for searching through one or more databases (201, 203, 205). The apparatus includes at least one hard wired search engine (904) arranged to receive the key and one or more programmable search parameters and to search through the data base for an address entry corresponding to the received key. The search is alterable by the programmable search parameters. Methods for searching through multiple data bases (100) are also disclosed.

WO 01/20501 A1

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

SEARCH ALGORITHM METHODS AND APPARATUS

BACKGROUND OF THE INVENTION

The present invention relates generally to search algorithms and data base structures that are suitable for such search algorithms. More particularly, the search algorithms and data base structures of the present invention may be implemented within a packet interface device.

Several types of data base structures are available today for storing large amounts of digital information. Accordingly, a large number of search algorithms have been developed to search the many types data base structures. Typically, a data base will include data entries, and each data entry is associated with an address entry that is associated with a key. The search algorithm is used to search the data base for a particular key. That is, a key is input into a search algorithm; the search algorithm matches the input key to a corresponding address entry within the data base; and the data entry corresponding to the matched address entry is output as the result of the search.

There are several different types of data base structures and corresponding search algorithms. For example, a binary search algorithm may be used to search the data base, and is well known to those skilled in the art. When a binary search is implemented, the data base address entries must be arranged to facilitate a binary search. That is, at least some of the address entries must have a link to a higher value address entry and a link to a lower value address entry. Thus, when a key is compared

to a particular address entry and there is no match, the search algorithm may continue the search by going to the higher or lower value address entry, depending on whether the key has a higher or lower value than the current address entry.

Although conventional search algorithms and data base structures work well for certain applications, they are not adequate for other applications. For example, when data is streaming through a data processing device, the search algorithm implemented by such processing device must perform required searches on the stream data at a rate that is at least as fast as the data rate. In certain packet switching systems, data flow rates are reaching tens of gigabits per second. Software based search algorithms typically cannot keep up with these flow rate speeds.

Thus, some switching device designers have developed hardware based search algorithms. Hardware search algorithms are custom designed circuits that perform a particular type of search algorithm. However, these hardware based search algorithms lack run time flexibility. In other words, the algorithm cannot be changed on a packet by packet basis. Many of these hardware based algorithms cannot even be statically changed, *e.g.*, not during run time.

In view of the foregoing, there is a need for improved data base structures and search methods that provide greater efficiency without sacrificing flexibility.

SUMMARY OF THE INVENTION

Broadly speaking, the present invention fills these needs by providing a search apparatus that implements a set of search algorithms and data base types. The search apparatus receives one or more programmable parameters for selecting a particular search algorithm, how it is implemented, and a particular data base type.

In an apparatus aspect of the invention, the set of algorithms are implemented by hardwired devices, such as state machines, that receive programmable parameters for selecting and/or modifying an algorithm. The advantage of hardwired devices is the speed, and the advantage of parameter based algorithm selection is flexibility. Also, it appears that there is significant "*time slack*" available in hard wired devices so that it is possible to parameterize them without significantly affecting performance. In one embodiment, this parameterization is controlled by micro code, and one gets the best of both worlds (performance, as well flexibility).

In a specific embodiment, an apparatus is disclosed that includes a master machine (referred to as micro engine) that is micro code based and a slave machine (referred to as macro engine) that consists of a set of state machines, each of which specializes in doing a particular type of search. The micro engine provides full flexibility of application specific searches, while the macro engine provides high performance for well defined applications. In another preferred embodiment, the macro engine also allows significant programmable parameterization, which is controlled by the micro engine.

In one embodiment, an apparatus for searching through a database is

disclosed. The apparatus includes at least one hard wired search engine arranged to receive the key and a programmable search parameter and to search through the data base for an address entry corresponding to the received key. The search is alterable by the programmable search parameter.

In an alternative embodiment, the apparatus includes a plurality of hard wired search engines arranged to perform one of a plurality of search algorithms for searching through a data base for a key. At least one of the search algorithms are alterable by the programmable search parameter. The apparatus further includes a key decoder arranged to receive the key and the programmable search parameter and to selectably output the key to one of the search engines based on the programmable search parameter or the key.

In another embodiment, the hard wired search engine is operable to search a plurality of data bases that each have a plurality of linked address groups that each have a plurality of address entries and a plurality of data entries that each correspond to a selected address entry. The search engine is arranged to sequentially compare a key to the address entries of a current address group and obtain a data entry corresponding to the address entry that matches the key. The obtained data entry is related to the key, and the search engine is also arranged to compare the key to the address entries of a next address group when the key does not match any of the address entries of the current address group.

In another embodiment, the search engine is operable to search the plurality of data bases even when each data base has a different number of address groups. In yet another aspect, the search engine is operable to search the plurality of data bases even

when each data base has a different number of address entries per address group.

In another embodiment, the plurality of data bases each further include a plurality of anchors that each reference a selected one of the address groups and the search engine is operable to obtain a first address group through a selected anchor. In a preferred embodiment, the anchor corresponds to a portion of the key. In another embodiment, the search engine is operable to search the plurality of data bases even when each data base has differently sized anchors or each of the anchors has either a 16 bit or 32 bit size.

In another aspect, an apparatus for searching through a database to compare received keys to a plurality of address entries is disclosed. Each address entry has a corresponding data entry. The apparatus includes a first state machine arranged to perform a first search of a first search type and a second state machine arranged to perform a second search of a second search type. The apparatus further includes a key decoder arranged to receive a key and a plurality of programmable search parameters and to selectably output the key to one of the state machines based on the programmable search parameters and/or the received key. The search performed by the selected state machine is based at least in part on the programmable search parameters.

In an alternative embodiment, the first search type is selected from a group consisting of a linked list search, a ranged based search, and a longest prefix search. In yet another embodiment, the second search type is selected from a group consisting of a linked list search, a ranged based search, and a longest prefix search, wherein the first search type differs from the second search type.

In another aspect, the invention pertains to a data base that includes a plurality of linked address groups that each have a plurality of address entries and a plurality of anchors that each reference a selected one of the address groups. The data base also includes a plurality of data entries that each correspond to a selected address entry. When a key matches a selected one of the address entries, a data entry corresponding to the matched address entry may be obtained. The data entry is related to the key.

In another aspect, the invention pertains to a method for searching through an address entry database for one or more portions of a key. The address entry group database includes a plurality of address entry groups each having a plurality of address entries. A current address group is searched for a current address entry that matches a current slice of a key. When a current address entry that matches the current slice is not found within the current address group, it is indicated that there is not a match. When a current address entry that matches the current slice is found within the current address group, a next address group associated with the current address entry of the current address group is searched for a next address entry that matches a next slice of the key when the next address group is valid. When a next address entry that matches the next slice is not found within the next address group after searching the next address group, it is indicated that there is a match corresponding to the current address entry. Each of the address groups have an associated current match length current indicating what length of the key must match the corresponding address entry, and a first address entry has a first current match length that differs from a second current match length of a second address entry.

In another embodiment, the invention pertains to a database configured for a

longest prefix search for a received key. The data base includes a plurality of linked address groups each having a plurality of address entries. Each address entry includes a match pattern that is comparable to a corresponding slice of the received key, a current match length that indicates how many bits of the corresponding key slice must match the match length, and a next group pointer. The data base also includes a plurality of data entries each associated with one of the address entries.

These and other features and advantages of the present invention will be presented in more detail in the following specification of the invention and the accompanying figures which illustrate by way of example the principles of the invention.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention will be readily understood by the following detailed description in conjunction with the accompanying drawings, wherein like reference numerals designate like structural elements, and in which:

Figure 1 is a flowchart illustrating a process for searching one or more databases in accordance with one embodiment of the present invention.

Figure 2 is a diagrammatic representation of a database in accordance with one embodiment of the present invention.

Figure 3 is a diagrammatic representation of an address entry group that may be utilized for a linked list search in accordance with one embodiment of the present invention.

Figure 4 is a flowchart illustrating a linked list search in accordance with one embodiment of the present invention.

Figure 5 is a diagrammatic representation of an alternative address entry group that may be utilized for a range based search in accordance with one embodiment of the present invention.

Figure 6 is an illustration of possible communication paths within a network system in accordance with one embodiment of the present invention.

Figure 7 is a diagrammatic representation of an alternative address entry group that may be utilized for a longest prefix search in accordance with one embodiment of

the present invention.

Figure 8 is a flowchart of a longest prefix search in accordance with one embodiment of the present invention.

Figure 9 is a diagrammatic representation of a search engine for implementing the search algorithms of the present invention in accordance with one embodiment.

Figure 10 is an example implementation of the macro search engine of Figure 9 in accordance with one embodiment of the present invention.

Figure 11 represents a state machine that implements the key decoder and latch of Figure 10 in accordance with one embodiment of the present invention.

Figure 12 represents a state machine that implements the linked list block of Figure 10 in accordance with one embodiment of the present invention.

Figure 13 represents a state machine that implements the memory arbitrator of Figure 10 in accordance with one embodiment of the present invention.

Figures 14A and 14B represent a state machine that implements the range based block of Figure 10 in accordance with one embodiment of the present invention.

Figure 15 represents a state machine that implements the longest prefix block of Figure 10 in accordance with one embodiment of the present invention.

DETAILED DESCRIPTION OF SPECIFIC EMBODIMENTS

Reference will now be made in detail to specific embodiments of the invention. While the invention will be described in conjunction with specific embodiments, it will be understood that it is not intended to limit the invention to the described embodiments. On the contrary, it is intended to cover alternatives, modifications, and equivalents as may be included within the spirit and scope of the invention as defined by the appended claims. In the following description, numerous specific details are set forth in order to provide a thorough understanding of the present invention. The present invention may be practiced without some or all of these specific details. In other instances, well known process operations have not been described in detail in order not to unnecessarily obscure the present invention.

Figure 1 is a flowchart illustrating a process 100 for searching one or more databases in accordance with one embodiment of the present invention. Initially, a key and its associated parameters are received in operation 102. One or more search algorithm(s) are then selected for the received keys based on the associated parameters in operation 104. In one embodiment, one or more specific search algorithm(s) may be selected from a set of search algorithms. At least one of these search algorithms may be implemented by one or more hard wired search engines having programmable search parameters.

In operation 106, the selected search algorithm(s) are then performed on the received key based on the associated parameters. That is, particular aspects of the selected search algorithm(s) are configurable based on the associated parameters.

Said in another way, a first set of parameters will result in a selected search algorithm being performed in a first way, and a second of parameters will result in a selected search algorithm being performed in a second way. The search process 100 then ends.

The search engines of the present invention may perform searches within various types of data bases. For example, the search engines may implement a first search algorithm on a first type of data base and a second search algorithm on a second data base. Additionally, for each type of search algorithm, the search engines may utilize a single data base or a plurality of data bases. By way of a final example, the search engines may utilize various configurations of a same type of data base (*e.g.*, linked list data base) for a particular type of search algorithm.

Figure 2 is a diagrammatic representation of a database 200 in accordance with one embodiment of the present invention. The database includes an anchor database 201, an address entry database 203, and a data entry database 205. The anchor database 201 includes a list of pointers 207. Each pointer 207 of the anchor database 201 references a particular group of address entries 210 within the address entry database 203. The anchor gives a starting place for the search algorithm. That is, each anchor references a first address group of a linked list to be searched.

Each address group contains a plurality of address entries. An individual address entry within an address entry group 210 references a particular data entry (e.g. via pointer 209) within data entry database 205. An address entry is typically compared to one or more portions of the received key. When a match occurs between a particular address entry and the received key, the referenced data entry may be retrieved via pointer 209 from data entry database 205.

The data entry is in any suitable format for storing information for later retrieval by a search algorithm. For example, within the context of a network system for transmitting data packets, the data entry may include information for the routing of data packets. By way of specific example, the data entry may include a destination, quality of service (QOS) information, and/or authentication parameters.

The address entry generally functions as an index to a particular data entry. For example, the address entry may correspond to a particular format of destination address that is compared to the received key. In other words, a particular destination address having a first format is received as part of the key, and is matched to a corresponding address entry having the same first format. The corresponding address entry is then used to fetch an associated data entry that represents the destination address in a second format.

The received key may have any suitable structure for facilitating a search. For example, the received key may include one or more portions that may be compared to one or more address entries. The received key may also include any suitable parameters (which are also referred to as "key attributes") for controlling how the search is performed. In one embodiment, the received key structure includes the following bits:

key type	4 bits
sub key type	4 bits
key value	size depends on key type & subtype
initial address value	24 bits
key attributes	24 bits
key mask	size depends on key type & subtype
initial address base	24 bits
data entry base	24 bits
address entry base	24 bits

Table 1

The key type defines the type of data base to be searched (*e.g.*, linked list or range based data base). The sub key type may be used to further define the type of data base. For example, the sub key type may specify one or more data bases from a plurality of linked list data bases. Additionally the sub key type may also specify whether the maximum size of the key value is one or two memory words (*e.g.*, 64 or 128 bits in length). The key value is the field that is compared to address entries. (The key value is also referred to as the "key"). The key value's size depends on the particular database being used. Thus, the key value's size depends on the key type and key subtype.

The initial address value may include a hashed value or any other portion of the key or database, including a constant value. The key attributes (also referred to as key parameters) includes other programmable fields that are selectable to control how the search is implemented or how the data base is structured. Example key attributes are further described below with reference to the descriptions of the three described search types: linked list, range base, and longest prefix search. The key mask may be used to mask out certain bits of the key value during the comparison. In other words, the mask specifies which bits of the key value are not to be compared to the corresponding bits of the address entry.

The initial anchor base may be used to access different anchor databases. In other words, the initial anchor value and the initial anchor base are both used to access a particular anchor database. In one embodiment, the initial anchor base is added to the initial anchor value to obtain a starting address group. Likewise, both of the data

entry base and the address entry base may be used to specify a particular entry. For example, two different address entry bases may be added to a same address entry to obtain two different data entries. Likewise, the data entry base may be added to a data entry pointer to obtain different data entries with the same data entry pointer.

Figure 3 is a diagrammatic representation of an address entry group 210 that may be utilized for a linked list search in accordance with one embodiment of the present invention. As shown, the address entry group 210 includes a next group UP pointer 301, a next group DOWN pointer 303, an entry data pointer 305, address entry group control bits 307, and an array of address entries 309.

Each address entry of the array 309 may include a match value, which will be compared to the received key, and one or more key control bits. The term "address entry" is also used to refer to this match value, excluding any bits of the match value that are masked out. The key control bits may be utilized to indicate various custom definitions. For example, a key control bit may indicate whether the address entry is valid. An address entry may be invalid for any number of reasons, *e.g.*, the address entry may have been aged out or it may not have been learned. Another key control bit may indicate whether a portion of the received key is to be masked out before a comparison is performed. Yet another control bit may be used to indicate a specific destination for a packet, such as the CPU. By way of a final example, other key control bits may be used to indicate debugging features like drop or copy.

At least a portion of the received key is compared to one or more of the address entries of a first address entry group. If a match is not found between the key and any one of the address entries of the first address group, the key or key portion is

then compared to the first address entry (e.g. address entry 0) to determine a next address group. The next address group is then accessed via the next group UP pointer 301 or the next group DOWN pointer 303 based on the results of the comparison. However, if the key or key portion matches one of the address entries 309, the entry data for the matching address entry may be accessed via entry data pointer 305.

The entry data pointer 305 references the entry data that corresponds to the first address entry (*e.g.*, address entry 0). The entry data corresponding to the remaining addresses entries (e.g. address entries 1-3) immediately follows the first data entry. That is, the data entries that correspond to a particular address entry array 309 are arranged in the same order as the address entries within the array 309. This is done to save memory, otherwise, the address entries and corresponding data entries may be arranged in any suitable manner. For example, each address entry may have a corresponding entry data pointer.

The address entry group control bits 307 are optional and may be utilized for any suitable custom function. For example, a control bit may indicate whether to discard the received key if it matches the associated address entry. The control bits may also be used to route the packet to the CPU, irrespective of whether or not a match is found for the received key.

In one embodiment, some options are selected via the above described key parameters that are received as part of the key. In one embodiment, the key parameters include the following programmable fields for the linked list search:

maximum tree width	4 bits
maximum group depth	3 bits
anchor size	1 bit

mask on upper bits of initial address value	14 bits
data entry size	2 bits

Table 2

The maximum tree depth indicates how many address groups are possible in a link. The maximum group width indicates how many address entries are within each address group. The anchor size indicates the size of the anchor value. For example, the anchor may be 16 bits or 32 bits in length. The mask may be utilized to mask out particular bits of the first address value such that the masked bits are not used for addressing the memory. The data entry size may indicate the size of the data entry.

Other programmable options may also be selectable for the linked list search, as well as for the range based and longest prefix searches. For example, the bases of the anchor area, address entry area, and data entry area may also be selectable. The maximum key size may also be selectable, *e.g.*, a 1 or 2 memory words. The basis for determining whether the next address group is obtained from the UP or DOWN pointer may also be selectable. In one embodiment, a full address entry or portion (or some other value) may be used as the UP/DOWN detection basis.

Figure 4 is a flowchart illustrating a linked list search in accordance with one embodiment of the present invention. Initially, an anchor is obtained in operation 402. The anchor may be obtained in any suitable manner. For example, the anchor may be obtained by modifying the initial anchor value by the anchor size and base of the anchor database 201. In one embodiment, the initial anchor value is divided by the anchor size and added to the anchor base to get the anchor address within the anchor database 201. Alternatively, a portion of the key (*e.g.*, an unhashed portion) may be used as an address or part of an address to get the anchor values within the

anchor database 201.

An address entry group that corresponds to the anchor is then obtained in operation 404. The address entries associated with the current address entry group are then searched to find a matching address entry in operation 406. It is then determined whether there is a match in operation 408. If there is a match, results are output indicating the data entry that is associated with the matching address entry in operation 410 and the linked list search 106 ends. The results may be in any suitable format for indicating the matching data entry. For example, the results may be in the form of the corresponding data entry pointer.

If there is no match via operation 408, the key is then compared to the first address entry within the current address entry group in operation 412. It is then determined whether the key has a higher or lower value then the first entry in operation 414. Of course, the key may be compared to any address entry within the address entry group. Additionally, the key may be compared to a special comparison value that is preset specifically for this purpose and does not necessarily correspond to a data entry. Generally, in that case the "valid" control bit of the address entry may be kept "0" to indicate that there is not a corresponding data entry.

If the key has a higher or same value than the first address entry, it is then determined whether there is a higher address entry group in operation 422. For example, it is determined whether the next group UP pointer 301 references a null value or a valid value. If there is a null value, results are output indicating that the key match has not been found in operation 424, and linked list search ends. If there is higher address entry group, the higher address entry group associated with the current

address entry group is obtained in operation 420 (e.g., via next group UP pointer 301). The address entries associated with the new current address entry group (e.g., the newly obtained higher address entry group) are then searched in operation 406.

However, if it is determined that the key has a lower value than the first address entry in operation 414, it is then determined whether there is a lower address entry group in operation 416. That is, it is determined whether the next group DOWN pointer 303 points to a valid address group. If there is not valid lower address entry group, results are output indicating that a key match has not been found in operation 424 and the linked list search ends. If there is a valid lower address entry group, the lower address entry group is obtained in operation 418 and the address entries of the lower address entry group are searched in operation 406.

The search continues until a match is found between a particular address entry of a particular address group and the received key. That is, the linked list of address groups is traversed through UP and DOWN pointers. After a match is found in operation 408, the results are output in operation 410. The results indicate the data entry associated with the matching address entry.

Figure 5 is a diagrammatic representation of an alternative address entry group 211 that may be utilized for a range based search in accordance with one embodiment of the present invention. Each address entry group 211 includes a plurality of policy entries 501. Each policy entry 501 includes, a data entry pointer 505, and address entry group control bits 507. Every other policy entry 501 (even numbers) includes a next group UP pointer 502 & a next group DOWN pointer 503. Of course, each policy entry include UP and DOWN pointers. These fields (*i.e.*, 501, 502, 503, 505,

and 507) function similarly to the same named fields of the linked list address entry group 201 of Figure 3.

Additionally, each policy entry 501 includes an address entry 509. Each address entry is arranged such that at least a portion of the received key may be compared to a range of match values. Any suitable arrangement of the address entry 509 may be implemented for comparing at least a portion of the key to a range of match values. For example, the address entry may simply provide a base value or starting value and a range value for comparison to the entire key. For example, the base value may have a value of 5, and the range value may have a value of 2. In this example, the key must be between the values of 5 and 7. Additionally, a range may be integer based as shown in the previous example or mask based. It may be possible to choose an integer or mask option on an address entry by address entry (or address entry group by address entry group) basis.

In the illustrated embodiment, the address entry 509 is in the form of a plurality of base and range sets. As shown, the address entry 509 includes a base of the destination IP address and a range of the destination IP address 509b. It may be determined whether the destination IP address field of the received key falls within the given range value specified by the base and range for the destination IP address (*e.g.*, 509a and 509b). Similarly, the address entry 509 includes a base of source IP address 509c, a range of source IP address 509d, a base of destination or source of the L4 address 509e, a range of the destination or source of the L4 address 509f, a base of L4 protocol 509g, a range of the L4 protocol 509h, a base of the TOS field 509i, and a range of the TOS field 509j. The corresponding fields within the received key may

then be checked to determine whether they follow within the indicated range values.

In one embodiment, some options are selected via above described key parameters that are received as part of the key. In one embodiment, the key parameters include the following programmable fields for the range based search:

use initial address (instead of L4) for getting anchor	1 bit
mask for L4 upper bits (if used)	5 bits
one deep entry (not full policy)	1 bit
use source IP for direction	1 bit
burst transfer count limit	12 bits
anchor size	1 bit

Table 3

The initial address may be selected as the anchor pointer, instead of the L4 address. One or more bits of the L4 source or destination address may be masked if the L4 source or destination address is used to get the anchor pointer. A one deep entry policy may also be selected. That is, a single field with a corresponding range may constitute the address entry group. The source IP address (as opposed to the destination IP address) may be selected as the basis for obtaining the UP or DOWN address pointer. The burst transfer count limit may also be selected. For example, the total number of memory transfers per burst may be selected in multiples of the address entry group's size. The anchor size indicates the size of the anchor value.

Other options that are described above for the linked list search may also be implemented for the range based search, *e.g.*, bases of anchor area, address entry area and data entry area, key size, and the width of an address group. In one embodiment, the size of the key may be selectable only if a one deep policy is used.

Another type of search algorithm that may be selected is called the longest

prefix match search. This type of search may be especially useful within a network system. Figure 6 is an illustration of possible communication paths within such a network system in accordance with one embodiment of the present invention. As shown, a first node 601 may communicate with a second node 605 via path 603. The second node 605 is coupled to a plurality of hosts 607. The second node 605 is also coupled to a third node 611 via path 609. The third node is coupled to a plurality of hosts 613. The second node 605 may represent, for example, a routing device for a plurality of hosts (*e.g.*, hosts 607 and 613) at a particular company site. The third node 611 may represent a routing device for a subset of hosts (*e.g.*. hosts 613) within the particular company site.

In the illustrated example, data may be routed from the first node 601 to the hosts 607 through path 603. Likewise, data may be routed from the first node 601 to the hosts 613 via paths 603 and 609. A key may be associated with the data indicating whether to route the data to node 605 or node 611.

In one embodiment, N bits of the key may indicate a destination address that corresponds to the second node 605, and N+M bits of the key may indicate a destination address that corresponds to the third node 611. If the N bits of the key match the destination address for the second node 605, the data will be routed to the second node 605. However, if the N bits match the destination address of the second node 605 and the M bits match the destination address of the third node 611, the data will then be routed to the third node 611. In sum, the second node 611 is the destination address only if N+M bits of the received key match. Otherwise, if only N bits match, the second node 605 is utilized.

Figure 7 is a diagrammatic representation of an alternative address entry group 210'' that may be utilized for a longest prefix search in accordance with one embodiment of the present invention. Each address entry group 210'' includes a plurality of address entries 701. Each address entry 701 includes a next group pointer 702, a data entry pointer 703, a next match record number 705, a next match length 707, a current match length 709, and a match pattern 711. Each address entry group corresponds to a particular set of bits within the key. For example, an address entry group may correspond to the first eight bits of the key. Accordingly, the address entry group would include at most 256 address entries (i.e., 2^8).

The match pattern 711 is compared to the corresponding key bits. The current match length 709 represents the required number of bits within that key portion that must match. For example, although the first eight bits of a key may be compared to some address entries within a particular address group, an individual address entry may only require that a sub portion of the 8 bits of the key be compared. Preferably, the address entries 701 within a particular address entry group are arranged from highest to lowest valued current match length value 709. In the eight bit example, the address entry that require all 8 bits to match is listed first and compared first to the corresponding key portion. The address entry that requires the next highest number of matching bits (*e.g.*, 7 bits) of the key to match is listed next. The bits required for the comparison are left justified in one embodiment.

The next match length 707 indicates the length of the longest match pattern in the next address entry group. For example, if the next address entry group is utilized to compare the next 7 bits of the key, the next match length is set to 7. The next

match record number 705 indicates the number of address entries within the next address entry group.

In one embodiment, the next match record number 705 is also utilized to indicate the type of search that is to be performed on the next address entry group. When the next match record number 705 is set to zero, a look up type search is performed. Otherwise a sequential search is performed. A look up type search may be desired when many match patterns are available for a particularly sized slice of the key. For example, for an 8 bit comparison, a look up search may be performed when there are more than 4 match patterns 711 available. This way sequential search related delays may be avoided at the cost of more memory. If the next match record number 705 is set to zero, it is assumed that the next address entry group contains $2^{(\text{next match length})}$ address entries. In case an address entry is found via look up, the current match pattern is not used. Instead the current match length decides match or no match. For example, a zero value of the current match length indicates a match, while a non zero indicates no match.

The entry data pointer 703 references the data entry that corresponds to the address entry. If a match is found between the key portion and the match pattern 711, the data entry pointer 703 may be utilized to access the corresponding data entry. The next group pointer 702 references the next address entry group for comparing other portions of the key. Thus, if a match is found between the key portion and the match pattern 711, the next address group pointer 702 may be accessed to obtain the next address entry group.

As described above, some options are selected via the above described key

parameters that are received as part of the key. In one embodiment, the key parameters include the following programmable fields for the longest prefix search:

maximum tree width	4 bits
maximum group depth	3 bits
initial next pattern width	4 bits
initial record number	4 bits
width of first slice used	5 bits

Table 4

Additionally, any other suitable options may be selectable. For example, widths of subsequent slices may be programmable. As a result, the number of address entries is also definable (as 1 through $2^{\text{slice width}}$ address entries). As described above, the comparison method is also programmable at each level of the tree, *i.e.,* look up or sequential comparison.

Figure 8 is a flowchart of a longest prefix search 106′ in accordance with one embodiment of the present invention. Initially, one or more width value(s) for the key's one or more key slice(s) are obtained in operation 801. The width value of the key slice corresponds to how many bits of the key will be utilized during the comparison for a particular address entry group. For example, a first slice width value of eight for a first address entry group indicates that eight bits of the key will be compared to the corresponding field of the address entries of the first address entry group. A second width value of seven indicates that seven bits of the key will be compared to the corresponding field of the address entries of the second address entry group in the search sequence.

Optionally, the anchor may also be obtained in operation 802. The first address entry group is then obtained that corresponds to the anchor in operation 803.

The address entries associated with the first address entry group are then searched for a match between the first key slice and the match patterns of the address entries in operation 805. It is then determined whether there is a match in operation 807. If there is no match, results are output indicating that the key match has not been found in operation 809 and the longest prefix search 106' ends.

If there is a match, it is then determined whether there is a next address group in operation 810. If there is not a next address entry group, results are output that are associated with the associated data entry of this address entry in operation 819. If there is a next address entry group, it is obtained in operation 811. The address entries associated with the next address entry group are then searched for a match between the second key slice and the match patterns of the next address entry group in operation 813. It is then determined whether there is match in operation 815. If there is no match, results are output associated with the last matching address entry in operation 819. These results correspond to the address entry that matches the highest number of bits of the key slice.

If there is a match, it is then determined whether there is next address entry group in operation 817. If there is no next address group, the results are output that are associated with the last matching address entry in operation 819. If there is a next address group, the next address entry group associated with the matching address entry is then obtained in operation 811. The next prefix search continues until there is no match within a particular address entry group or the end of the address entry group chain is reached. Results are output that correspond to the data entry associated with the last matching address entry.

Any suitable combination of hardware and software may utilized to implement the search engine of the present invention. Preferably, the search engine algorithms are at least in part implemented within hardware so as to maximize search speed. The search engine hardware is also configurable to receive variable search parameters or options, *e.g.*, as described above.

Referring next to Fig. 9, one embodiment of a search engine 900 for implementing the above described processes of Figures 1 through 8 will be described. As shown, the search engine 900 includes an input key scheduler 902, two search engines 904a and 904b, an entry cache controller 908, an accounting and learning controller 925, address and data modules 910 and 912, an external memory controller 914, an aging controller 920, local memory resources 918, an external access port 916, and an output result scheduler.

In general terms, a key 901 may be received into the search engine 900 and used by the search engine 900 to generate and output an output result 903. The search engine 900 performs one or more searches based, in part, on the received key. The input key scheduler 902 receives a key or key portion 901. The input key scheduler 902 also checks entry cache control 908 for previously input keys or key portions. The entry cache control 908 may contain previously obtained search results for previously input keys or key portions. In other words, a key or key portion that is identical to the current key or key portions may have been previously received by the search engine 900 and stored within cache 908, along with associated output results.

The key or key portion may then be passed to the micro search engine 904a. The micro search engine is a micro-code based engine that is configured to control the

macro search engine 904b. The macro search engine 904b is arranged to search through a plurality of address entries by varying the address signals. At the direction of the micro search engine 904a, the macro search engine 904b searches for a particular address entry that corresponds to one or more portions of the received key. When a corresponding address entry is found, it's associated data entry may be retrieved (if one exists for the corresponding address entry) via a data pointer that is located with the corresponding address entry. Alternatively, the data entry may be directly located with the corresponding address entry without using a data pointer, for example. The retrieved data entry may then be used to generate at least a part of the output result 903, which is output from the search engine 900 via the output scheduler 922.

The micro search engine 904a may be programmed in any suitable manner to meet the requirements of the particular application. For example, the micro search engine may be programmed with a particular search algorithm. Alternatively, the micro search engine 904a may hand the key to the macro search engine 904b so that the macro search engine perform the search. The micro search engine 904a may also be programmed to complete a search partially performed by the macro search engine. The micro search engine is generally responsible for examining the key contents and/or database to determine and calculate routing parameters, such as quality of service. The micro search engine may also implement other functions, such as learning and accounting. In one embodiment, the micro search engine is equipped with specialized hardware for performing long string searches.

The macro search engine 904b is hard wired to provide rapid searches.

Although the macro search engine is hard wired, it still flexibly provides for a number of programmable search options, as outlined above. Additionally, as described above, a particular search algorithm may be selected to be implemented by the macro engine and/or micro engine. Also, parameters of the search algorithm may be configured (*e.g.*, through received key parameters). For example, a linked list search, range based search, and/or longest prefix search may be selected for a given key. For each type of search, different database configurations may be implemented, such as different anchor databases and/or address entry group databases. For instance, the size of the anchor pointer may be configured. Additionally, the width of each address entry group (*e.g.*, the number of address entries per group) may also be selected.

As described above, the received key may include a type and subtype to define the search to be performed by the search engine 900. Of course, any number of fields may be used to define the search, such as a single field or multiple fields. In this embodiment, the type field is used to indicate a type of search algorithm, such as a linked list or range based search, and the subtype field indicates which database to search. For example, the type and subtype fields may together indicate which database to search, and how to search it. Each database is constructed for a particular search algorithm. However, since more than one database may be used for each search algorithm, the subtype is used to specify the database.

Address module 910 generates address signals for accessing an external memory (not shown). The external memory device may contain one or more databases that are accessed by the engine 904. Alternatively, there may be more than one memory device with each containing one or more databases that are accessible by

the engine 904. For example, the address signals can be used to access a specific address entry within the database(s).

The data module 912 may be used to read data (*e.g.*, an address entry or data entry) from the database(s). For example, when the engine 904 outputs a specific set of address signals to the address module 910 to access a data pointer, the data pointer value value is output from one of the database(s) of the memory device to the data module 912. The engine 904 may then change the address signals to the address module 910 to correspond to the data pointer value and access the data entry from the data module 912.

The data module 912 may also be configured to perform a write to one of the databases. The write may be initiated in any number of circumstances. For example, a write operation may be performed during a learning function, wherein new entries are added to the database(s). By way of another example, a write operation may be performed to "age" the data entries. In other words, each time a address entry is accessed (or "hit"), it is marked to indicate use. The address entries may then be periodically checked for marks, and address entries and associated data pointers and data entries that are not used for a predefined period of time may then be purged from the database(s). The learning and aging functions are further described below.

The external memory controller 914 may be used to generate additionally required control signals for accessing the memory device. For example, the memory signals may include an address strobe and/or chip select signal to enable the memory device, a read and write enable signal(s) to enable a read or a write to the memory device, and one or more byte enable signals to enable access to specific bytes of data

within the database(s). In sum, the external memory controller 914 provides control signals to the memory device; the address module 910 provides address signals to the address bus of the memory device; and the data module 912 provides data signals to the data bus of the memory device and reads data signals from the memory device.

As described above, the search engine 900 may also include learning capabilities for writing new data entries to one or more database(s). The learning function may be implemented by the engine 904 itself or by the CPU through an external access port block 916. Specifically, the CPU may write a new entry to the memory device through the external access port block 916 and data module 912.

The local memory resources 918 may be used for any suitable storage purposes. For example, one or more of the registers may be programmable and used to alter search algorithms. Additionally, some of the registers may be used as intermediate storage registers to temporarily hold information obtained from the database(s). The stored information is released when the engine 904 completes its searches for a particular set of keys and then used to create output results 903. These resources 918 may also be used as a scratch pad by the micro engine 904a.

The aging controller 920 may delete inactive address entries and associated data pointers and data entries from the database(s). That is, data entries are purged if they are not used by the engine 904 to generate the output results after a predefined period of time. The aging controller 920 works with the input key scheduler 902 and engine 904 to mark data entries within the database(s) that are found using the received key. The aging controller 920 includes a timer that is used to periodically clear the marks from the data entries. The data entries may then be marked again

when accessed by the engine 904. After the predefined time period. the data entries are checked to determine whether any entries are obsolete. In other words, if a particular data entry has not been marked within the predefined time period, it is purged from the data base by the engine 904 via the data module 912. In sum, data entries that are not used or accessed for long periods of time may be deleted from the database(s).

Alternatively, the above described marking approach may be replaced by implementing an activity count. The aging controller reduces the activity count by one at a predefined interval. When the count reaches zero, the aging controller sends a "delete key" instruction to the input scheduler, which instruction is then sent to the micro engine 904a. The micro engine 904a then deletes the entry. The aging controller may also be programmed to inform the CPU of this event (*e.g.*, the activity counter has reached zero). The CPU may be informed, for example, via an interrupt, instead of generating a "delete key" instruction.

The output results 903 are output through the output scheduler 922 to a receive FIFO, for example. The output scheduler 922 may also include storage capabilities for accumulating output results until the receive FIFO is ready to accept the result. The output scheduler 922 may also include a timer for determining when to output the output results to the receive FIFO.

The macro search engine 904b may include any suitable combination of hardware and software arranged to quickly perform a plurality of at least partly preconfigured search algorithms. In one embodiment, the search engine includes a plurality of search blocks that are each preconfigured to perform a particular search

algorithm. For example, a first search block is preconfigured to perform a linked list search, and a second search block is preconfigured to perform a range based search. Although the search blocks are partly preconfigured, the search blocks may also accept a set of programmable search parameters. The search parameters may indicate various programmable features of the search algorithm itself, as well as the type and configurations of the data bases that are searched.

Figure 10 is an example implementation of the macro search engine 904b of Figure 9 in accordance with one embodiment of the present invention. As shown, the macro search engine 904b includes a key decoder and latch 1001, three key buffers 103, three search blocks 1005, a memory arbitrator 1009 , and a result output block 1007. The key and associated key parameters are received into key decoder and latch 1001. The key decoder 1001 is arranged to determine where to forward the key and to request a specific type of search algorithm for the key. As shown, the key decoder 1001 may forward the key and key parameters to one of three buffers 1003. A first buffer 1003a is coupled with linked list block 1005a; a second buffer 1003b is coupled with longest prefix block 1005b; and a third buffer 1003c is coupled with range block 1005c. Key decoder 1001 requests either a linked list search from linked list block 1005a; a longest prefix search from longest prefix block 1005b; or a range based search from range block 1005c.

Each of the search algorithm blocks (i.e., 1005) is arranged to communicate with memory arbitrator 1009. Memory arbitrator 1009 is configured to access an external memory interface (e.g., external memory controller 914, address module 910 and data module 912 of Figure 9). Each search engine algorithm is also arranged to

output a search result to result output block 1007. This search result is then passed to the micro search engine 904a of Figure 9.

Figure 11 represents state machine implementations of the key decoder and latch 1001 of Figure 10 in accordance with one embodiment of the present invention. During idle state 1102 search request signals are set. For example, particular search request signals are set based on the key parameters. In one embodiment, if the key type indicates that a linked list database is to be used, a channel request for a linked list data base is set.

When a new and valid key is received, an acknowledgement of the key is indicated in state 1104. After the key is acknowledged, the state machine 1001 returns to the idle state 1102. If a new and invalid key is received, an error is indicated in state 1106. The state machine is then returns to the idle state 102. Invalidity may be determined based on whether the key contents are arranged within one of a plurality of patterns. By way of specific example, it may be determined whether the received key has the expected key type and subtype parameter values (*e.g.*, for the search algorithm that is selected).

Figure 12 represents state machine implementations of the linked list block 1005a of Figure 10 in accordance with one embodiment of the present invention. The linked list block 1005a includes two state machines, a main state machine 1201 and a comparator state machine 1203. The main state machine 1201 generally sets up signals for accessing particular address entries of a particular address group and requesting a comparison between the key and accessed address entries. The comparator is generally responsible for initiating a comparison between the key and

accessed address entries. The comparator also loads a next address group if no match is found (if a next address group is present). Of course, these tasks may be divided between any number of state machines.

Referring to the main state machine 1201, a set up procedure for the linked list block occurs in state 1202. This set up procedure may include setting control signals based on programmable search parameters. For example, a request64 signal is set depending on whether a 64-bit linked list search or a 128-bit linked list search has been requested. In the illustrated embodiment, the request64 signal is set based on the sub key type described with reference to Table 1. That is, the state machine for the linked list block 1005a accepts a key that is 64 or 128 bits in length. A transfer count signal is also set to the value of the maximum width specified by the key attributes, which specifies the maximum number of address entries per address group. A depth signal is also decremented to indicate how many address groups have been (or are about to be) processed (e.g., compared with the key). An error may be indicated when the number of processed address groups exceeds the specified maximum depth.

After a request for a linked list search is received from the key decoder 1001, signals for accessing memory are then set up in state 1204. For example, a channel request signal that is output to the memory arbitrator 1009 is activated (*e.g.*, set to a high value) until the memory arbitrator 1009 acknowledges receipt of the channel request. Additionally, a last request signal that indicates whether the a last block of data is being requested from the memory arbitrator 1009 is set. The last request signal is utilized in a later state to process the last block of data that is still pending from the memory. An address strobe signal is also activated until a memory acknowledgement

signal is received from the memory arbitrator 1009.

After the memory ready signal is activated by the memory arbitrator, the signals for loading the anchor are then set in state 1206. After the anchor is loaded, signals for accessing the memory (e.g., referenced by the anchor) are then set in state 1208. When the memory is ready, signals for accessing the address entries (e.g., of the address group obtained in the previous state) are set and a comparison is requested for an address entry and the received key in state 1210.

Referring to the comparator state machine 1203, comparisons between the received key and address entries of the currently loaded address group are initiated and analyzed. Comparison set up signals are set in state 1218. For example, a signal to initiate the loading of pointers to the next address group (*e.g.*, the UP or DOWN pointer) is activated. The next address group may be required if no address entries match the key within the current address group.

When a compare is requested, a comparison between the key and the address entries of the current address group is triggered in state 1220. The transfer count (which was initialized to the maximum width value) is decremented as each address entry is compared. If there is a match (e.g., between an address entry and key), a match found signal is set in state 1222. If there is not a match and the transfer count has reached zero, a next address group is then loaded (*e.g.*, through the UP or DOWN pointer) and it is indicated that the compare is over in state 1224.

Turning back to the main state machine 1201, if the match found signal is set, a search ready signal is activated in state 1212. As a result of a match being found, a

result is output. The micro search engine (e.g., 904a of Figure 9) takes the result that is output from the linked list block. The result may include a flag indicating that there was a hit. The state machine for the linked list block then returns to idle state 1202.

If the match found signal is not set, signals are then set for requesting the last or only comparison in state 1216. This state may be provided to allow enough time (*e.g.*, clock cycles) for the comparison to finish outputting a result. If a match is found for the last comparison, a search ready signal is set in state 1212. If a match is not found, memory access signals are again set if a next address group is present in state 1214. This determination may be based on a null signal. The null signal is set if the end of the linked list has been reached. If the null signal is not set, signals for accessing the memory (e.g., the next address group) are set in state 1208. If the null signal is set, the search ready is set in state 1212 and the search ends. For this case, the result output may include a flag indicating a miss.

Figure 13 represents state machine implementations of the memory arbitrator 1009 of Figure 10 in accordance with one embodiment of the present invention. In state 1302, signals for selecting a memory grant to either the linked list block 1005a, the longest prefix block 1005b, or the range block 1005c are preset based on values of the channel request signals from the respective blocks 1005. For example, if the channel request from the linked list block is set, signals for a linked list grant are set.

When a longest prefix search channel is requested, a longest prefix grant is performed in state 1304. This represents the highest priority grant although other priority sequences may be contemplated. When there are no more channel requests (longest prefix or otherwise), the state machine 1009 returns to presetting the grant

signals in state 1302.

The second highest priority is the linked list memory grant. As illustrated, when there is a linked list request and there is not a longest prefix request, a linked list grant is initiated in state 1304. When there are no more requests, the state machine 1009 returns to state 1302.

The lowest priority is the ranged-based grant. When a range request is received and there is no linked list request or longest prefix request, a range-based grant is initiated in state 1306. When there are no more requests and there is not a last range based request, the state machine 1009 returns to state 1302.

From the longest prefix grant state 1304, a linked list grant may be initiated in state 1304 when a linked list request is present and no longest prefix request is present. The linked list grant state 1304 may also be obtained from the range grant state 1306 when there is a linked list request and there is not any longest prefix request or range based request. This helps enhance overall performance.

A longest prefix grant state 1304 may be reached from the linked list grant state 1304 when there is a longest prefix request and there is no longer any linked list requests. The longest prefix grant state 1304 may also be reached from the range grant state 1306 when there is a longest prefix request and there is not a ranged based request.

The range based grant state 1306 may be reached from the linked list grant date 1304 when there is a range request and no longest prefix request, as well as no other linked list request. The range based grant state 1306 may also be reached from

the longest prefix state 1304 when there is a range request and there not a longest prefix request or linked list request.

Figures 14A and 14B represent state machine implementations of the range block 1005c of Figure 10 in accordance with one embodiment of the present invention. The state machine for the range based block 1005c shares many states in common with the state machine for the linked list block 1005a, as illustrated in Figure 12A. Signals for accessing a particular address group are set up in states 1402 through 1410. State 1416 includes setting signals for requesting a last or only compare, and state 1412 sets signals indicating that a search is ready.

The range based state machines also includes a state 1414 for setting memory signals for a next burst. This state configures signals to access one or more multiple address group entries during a single memory burst cycle. After the signals for accessing address entries and requesting a compare are set up in state 1410, a new burst may be set up in state 1414 (if a match has not been found and the burst is over but the comparison check is not over). The compare request is also withdrawn when leaving state 1410 (as well as when leaving state 1416). If a one deep policy is required, a new burst may also be set up in state 1414 after setting up signals for accessing an address group in state 1408. After a new burst is set up in state 1414, signals for accessing the next address group are then set up in state 1408 if the previous match is done and the comparison check is not complete and there was no match found. If a match is found or the check is over, a search ready signal is set in state 1412. The check is over, for example, when the predefined burst size runs out. The burst size may be defined any number of ways, such as by a received key

parameter. The burst size is loaded and decremented for each burst. When the burst size reaches zero, the check is over.

Turning to Figure 14B, comparison signals are set up in state 1416. For example, a flag to indicate whether the address entry is odd or even is set. When a compare is requested, signals are set up to access a 32-byte even address entry in state 1418. After a comparison is performed on the even address entry, signals are then set up to access a 32 byte odd address entry in state 1420 if the compare request is still active and a one deep policy is not required. If a 1-deep policy is required (*e.g.*, via the received key attributes), the state machine 1005c returns to setting up the comparison signals in state 1416. In a 1 deep policy, only a single field of a single address entry of each address entry group (*e.g.*, destination IP address range and base) is compared to a corresponding part of the key. The state machine 1005c also returns to setting up the comparison signal in state 1416 when the compare request is withdrawn.

In sum, the signals for accessing each 32 byte odd address entry are set up in state 1426, and the signals for accessing each 32 byte even address entry are set up in state 1418. In the illustrated embodiment, the UP or DOWN pointer to the next address pointer is loaded only when an even address entry is accessed since the odd address entries do not contain these pointers.

Figure 15 represents state machine implementations of the longest prefix block 1005b of Figure 10 in accordance with one embodiment of the present invention. The state machine for the longest prefix block 1005b shares many states in common with the state machine for the linked list block 1005a, as illustrated in Figure 12. Signals

for accessing memory and requesting a comparison are set up in states 1502 through 1508. State 1504 includes signals for setting up the comparison. For example, a depth signal is decremented to indicate how many address groups have or are about to be analyzed. State 1510 includes setting signals for requesting a last or only compare, and state 1514 includes indicating that a search is ready.

Comparison set up signals are set in state 1516. For example, a signal to initiate the loading of the first address group is activated. When a compare is requested, a comparison is triggered in state 1518. If any of the records (*e.g.*, address entries) match (*e.g.*, the required part of the match pattern of the address entry and the same part of the corresponding key slice), a match found signal is set in state 1520. If none of the records match, a next address group is then loaded and it is indicated that the compare is over in state 1522.

Traversal from state to state within the longest prefix search state machine differs from the linked list state machine. In the linked list case, the search continues on a mismatch, while in the longest prefix case, the search continues on a match. When a look up is indicated (*e.g.*, via a key attribute or address entry field) during the memory access state 1506, signals for requesting the only comparison are set in state 1510. In other words, a single comparison is performed. When memory signals for accessing a next record (*e.g.*, next address entry) are being set in state 1512, a null may be encountered when the end of the address entries is reached. Additionally, the end of the key may have been reached. If the end of the key or a null is reached or no match is found, the search ready signal is activated in state 1514 and the search then waits in the setup state 1502. If the end of the key or null has not been reached, the

signals for accessing memory (*i.e.*, of the next record) are set in state 1506 and the comparison continues.

Comparisons continue for each next record and key slice until the end of the key or the end of the records are reached or no match is found. The results are then output to the micro search engine 904a. The results may indicate the data entry or the data pointer of the last matching address entry or indicate that no match was found. For example, the results may reference a null. The last matching data entry corresponds to the address entries that matches the most bits of the key.

Although the foregoing invention has been described in some detail for purposes of clarity of understanding, it will be apparent that certain changes and modifications may be practiced within the scope of the appended claims. It should be noted that there are many alternative ways of implementing both the process and apparatus of the present invention. For example, although search algorithms were described in the context of network system, some of the search mechanisms may also be applied to any other type of search application. Accordingly, the present embodiments are to be considered as illustrative and not restrictive, and the invention is not to be limited to the details given herein, but may be modified within the scope and equivalents of the appended claims.

What is claimed is:

CLAIMS

1. An apparatus for searching through a database (201, 203, 205), the apparatus comprising a hard wired search engine (904) arranged to receive the key and a programmable search parameter and to search through the data base for an address entry corresponding to the received key, the search being alterable by the programmable search parameter.

2. An apparatus as recited in claim 1 further comprising:

a plurality of hard wired search engines arranged to perform one of a plurality of search algorithms for searching through a data base for a key, at least one of the search algorithms being alterable by the programmable search parameter; and

a key decoder arranged to receive the key and the programmable search parameter and to selectably output the key to one of the search engines based on the programmable search parameter or the key.

3. An apparatus as recited in claim 1 wherein the hard wired search engine is arranged to perform a search selected from a group consisting of a linked list search, a range based search, and a longest prefix search.

4. An apparatus as recited in claim 1 wherein the hard wired search engine is operable to search a plurality of data bases each having a plurality of linked address groups each having a plurality of address entries and a plurality of data entries that

each correspond to a selected address entry, wherein the search engine is arranged to sequentially compare a key to the address entries of a current address group and obtain a data entry corresponding to the address entry that matches the key, the obtained data entry being related to the key, the search engine also being arranged to compare the key to the address entries of a next address group when the key does not match any of the address entries of the current address group.

5. An apparatus as recited in claim 4 wherein the search engine is operable to search a selected data bases even when the data base has at least some differently sized address entries.

6. An apparatus as recited in claim 5 wherein the search engine is arranged to perform a longest prefix search.

7. An apparatus as recited in claim 4 wherein the search engine is operable to search the plurality of data bases even when each data base has a different number of address groups.

8. An apparatus as recited in claim 4 wherein the search engine is operable to search the plurality of data bases even when each data base has a different number of address entries per address group.

9. An apparatus as recited in claim 4 wherein the plurality of data bases each further include a plurality of anchors that each reference a selected one of the address

groups and the search engine is operable to obtain a first address group through a selected anchor.

10. An apparatus as recited in claim 9 wherein the anchor corresponds to a portion of the key.

11. An apparatus as recited in claim 9 wherein the search engine is operable to search the plurality of data bases even when each data base has differently sized anchors.

12. An apparatus as recited in claim 11 wherein a first anchor of a first database has a size of one memory word and a second anchor of the first data base has a size of two memory words.

13. An apparatus as recited in claim 11 wherein the search engine is operable to search the plurality of data bases even when each data base has a different number of anchors.

14 An apparatus as recited in claim 11 wherein each of the anchors reference a selected one of the address groups through an associated anchor pointer and the search engine is operable to search the plurality of data bases even when each data base has differently sized anchor pointers.

15. An apparatus as recited in claim 14 wherein a selected anchor pointer is

obtained by dividing an initial address value by an anchor size and adding an anchor base.

16. An apparatus as recited in claim 4 wherein the search engine is arranged to compare a selected one of the address entries of the address group being compared to the key to determine whether a lower address group or an upper address group is to be compared next to the key when none of the address entries match the key.

17. An apparatus as recited in claim 16 wherein the search engine is arranged to perform a linked list search.

18. An apparatus as recited in claim 4 wherein the search engine is arranged to compare a selected one of the address entries of the address group that is not being compared to the key to determine whether a lower address group or an upper address group is to be compared next to the key when none of the address entries match the key.

19. An apparatus as recited in claim 16 wherein the search engine is arranged to perform a range based search.

20. An apparatus as recited in claim 19 wherein the search engine is arranged to compare either a first or second one of the address entries of the address group being compared to the key to determine whether the lower address group or the upper address group is to be compared next to the key when none of the address entries

match the key.

21. An apparatus for searching through a database to compare received keys to a plurality of address entries, each address entry having a corresponding data entry, the apparatus comprising:

a first state machine arranged to perform a first search of a first search type;

a second state machine arranged to perform a second search of a second search type; and

an key decoder arranged to receive a key and a plurality of programmable search parameters and to selectably output the key to one of the state machines based on the programmable search parameters and/or the received key,

wherein the search performed by the selected state machine is based at least in part on the programmable search parameters.

22. An apparatus as recited in claim 21 wherein the first search type is selected from a group consisting of a linked list search, a ranged based search, and a longest prefix search.

23. An apparatus as recited in claim 22 wherein the second search type is selected from a group consisting of a linked list search, a ranged based search, and a longest prefix search, wherein the first search type differs from the second search type.

24. A data base comprising:

a plurality of linked address groups each having a plurality of address entries;

a plurality of anchors that each reference a selected one of the address groups; and

a plurality of data entries that each correspond to a selected address entry,

wherein when a key matches a selected one of the address entries, a data entry corresponding to the matched address entry may be obtained, the data entry being related to the key.

25. A data base as recited in claim 24 wherein at least some of the address groups are associated with an UP pointer that references an upper address group and a DOWN pointer that references a lower address group.

26. A data base as recited in claim 25 wherein the at least some of the address groups include a first address entry that has a value that if compared to the key indicates whether the lower address group or upper address group is to be compared next to the key.

27. A data base as recited in claim 24 wherein each address entry has a base value and a range value for a plurality of fields.

28. A data base as recited in claim 27 wherein the range value is either an integer type or a mask type.

29. A data base as recited in claim 28 wherein a first address entry has a range value of an integer type and a second address entry has a range value of a mask type.

30. A data base as recited in claim 24 wherein at least some of the address entries include a control field that is indicative of a characteristic of the corresponding address entry or its associated data entry.

31. A data base as recited in claim 30 wherein the control field indicates whether the corresponding address entry is valid or present.

32. A data base as recited in claim 30 wherein the control field indicates whether particular positions within the key should be masked off before a comparison is performed.

33. A data base as recited in claim 30 wherein the control field indicates debugging features.

34. A data base as recited in claim 33 wherein the debugging features are selected from a group consisting of sending a packet corresponding to the key to a CPU, discarding the packet, and copying the packet to the CPU.

35. A data base as recited in claim 24 wherein the data entries corresponding to a particular address group are obtainable from a first address entry within the particular address group.

36. A method for searching through an address entry database for one or more

portions of a key, wherein the address entry group database includes a plurality of address entry groups each having a plurality of address entries, the method comprising:

(a) searching a current address group for a current address entry that matches a current slice of a key;

(b) when a current address entry that matches the current slice is not found within the current address group, indicating that there is not a match;

(c) when a current address entry that matches the current slice is found within the current address group, searching a next address group associated with the current address entry of the current address group for a next address entry that matches a next slice of the key when the next address group is valid; and

(d) when a next address entry that matches the next slice is not found within the next address group after searching the next address group, indicating that there is a match corresponding to the current address entry,

wherein the address groups each have an associated current match length current indicating how what length of the key must match the corresponding address entry, a first address entry having a first current match length that differs from a second current match length of a second address entry.

37. A method as recited in claim 36 further comprising when a next address entry that matches the next slice is found within the next address group, defining the next address group as the first address group, the next address entry as the first address entry, and the next slice as the first slice, and repeating operations (c) through (e).

38. A method as recited in claim 37 wherein a size of the current slice differs from a size of the next slice.

39. A method as recited in claim 37, wherein the first and second current match lengths belong to a same address group.

40. A method as recited in claim 36 wherein the address entries of each address group are arranged in descending order by current match length.

41. A method as recited in claim 36 wherein a sequential type or a look up type is selectable for the searches.

42. A method as recited in claim 41 wherein a first search type selected from a sequential type and a look up type is performed on the current address group and another search type selected from a sequential type and a look up type is performed on the next address group.

43. A database configured for a longest prefix search for a received key, comprising:

a plurality of linked address groups each having a plurality of address entries, wherein each address entry includes a match pattern that is comparable to a corresponding slice of the received key, a current match length that indicates how many bits of the corresponding key slice must match the match length, and a next group pointer; and

a plurality of data entries each associated with one of the address entries.

44. A database as recited in claim 43 wherein the address entries of each address group are arranged in descending order of current match length.

45. A database as recited in claim 43 wherein each address entry also includes a next match length that indicates a length of a longest match pattern within a next address group associated with the next group pointer.

46. A database as recited in claim 43 wherein each address entry also includes a next match record number that indicates a type of search that is to be performed on a next address group associated with the next group pointer.

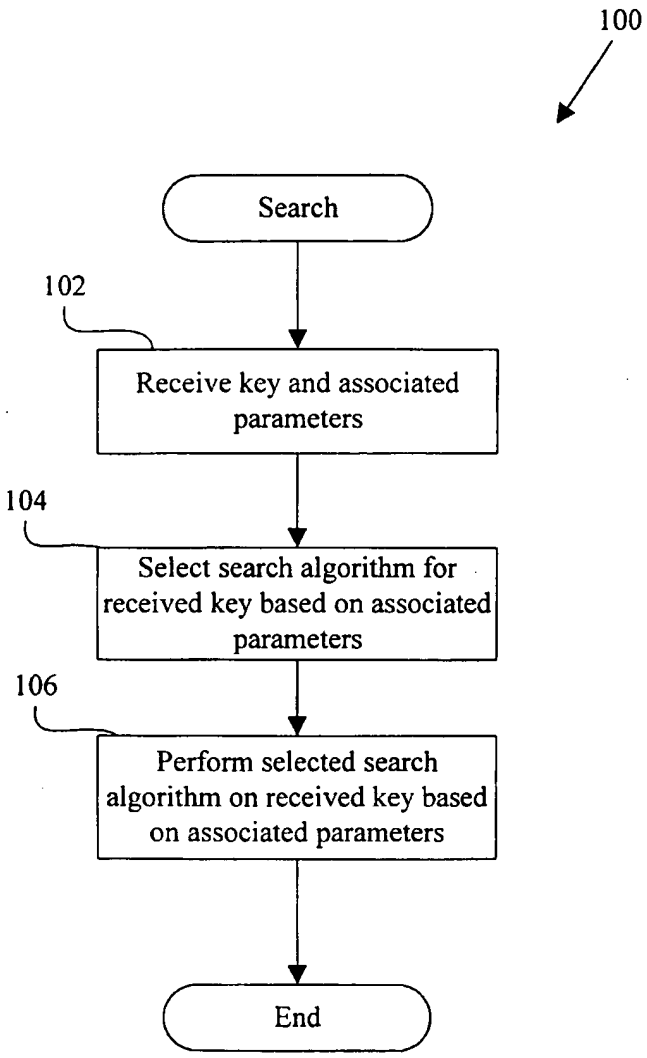

100
Search
102
Receive key and associated parameters
104
Select search algorithm for received key based on associated parameters
106
Perform selected search algorithm on received key based on associated parameters
End

Figure 1

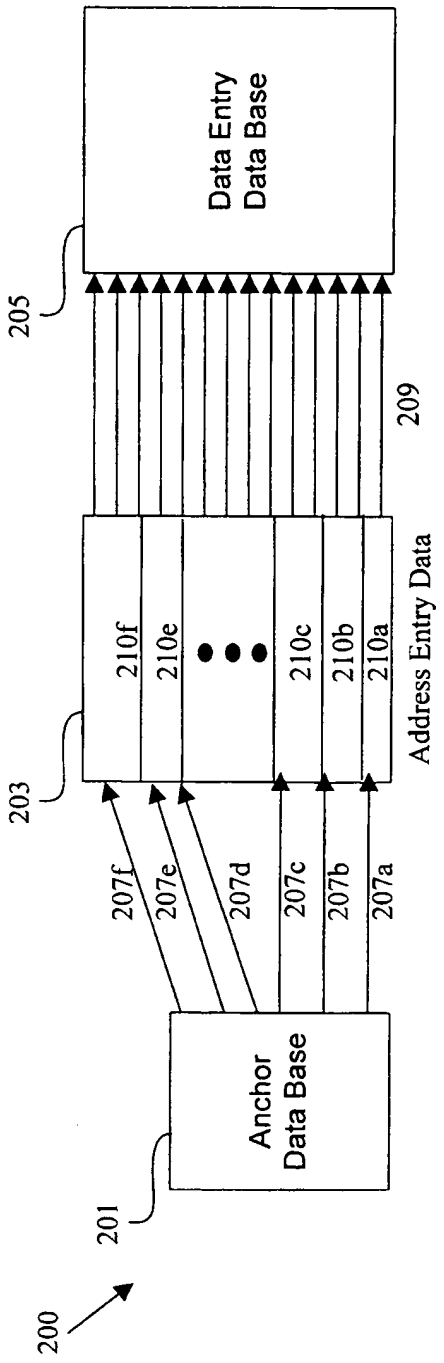


Figure 2

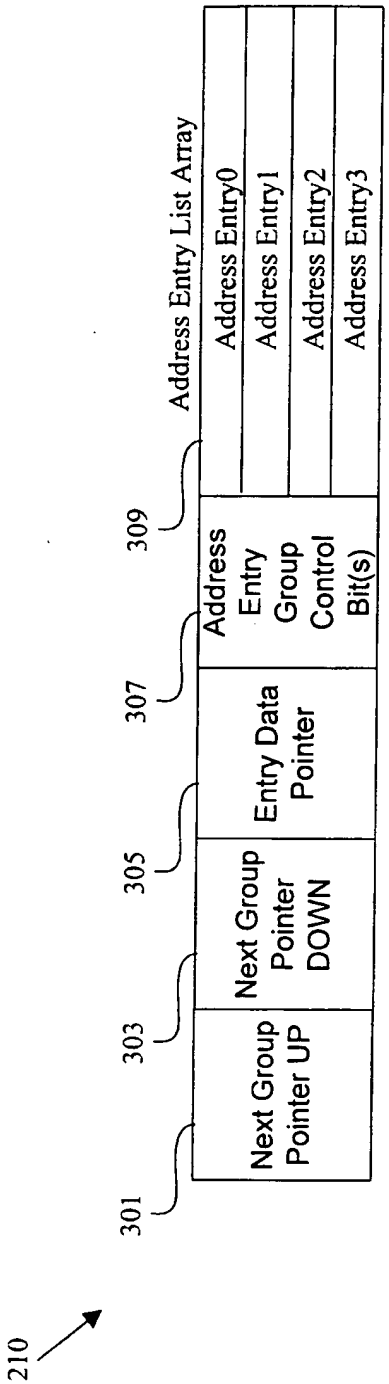


Figure 3

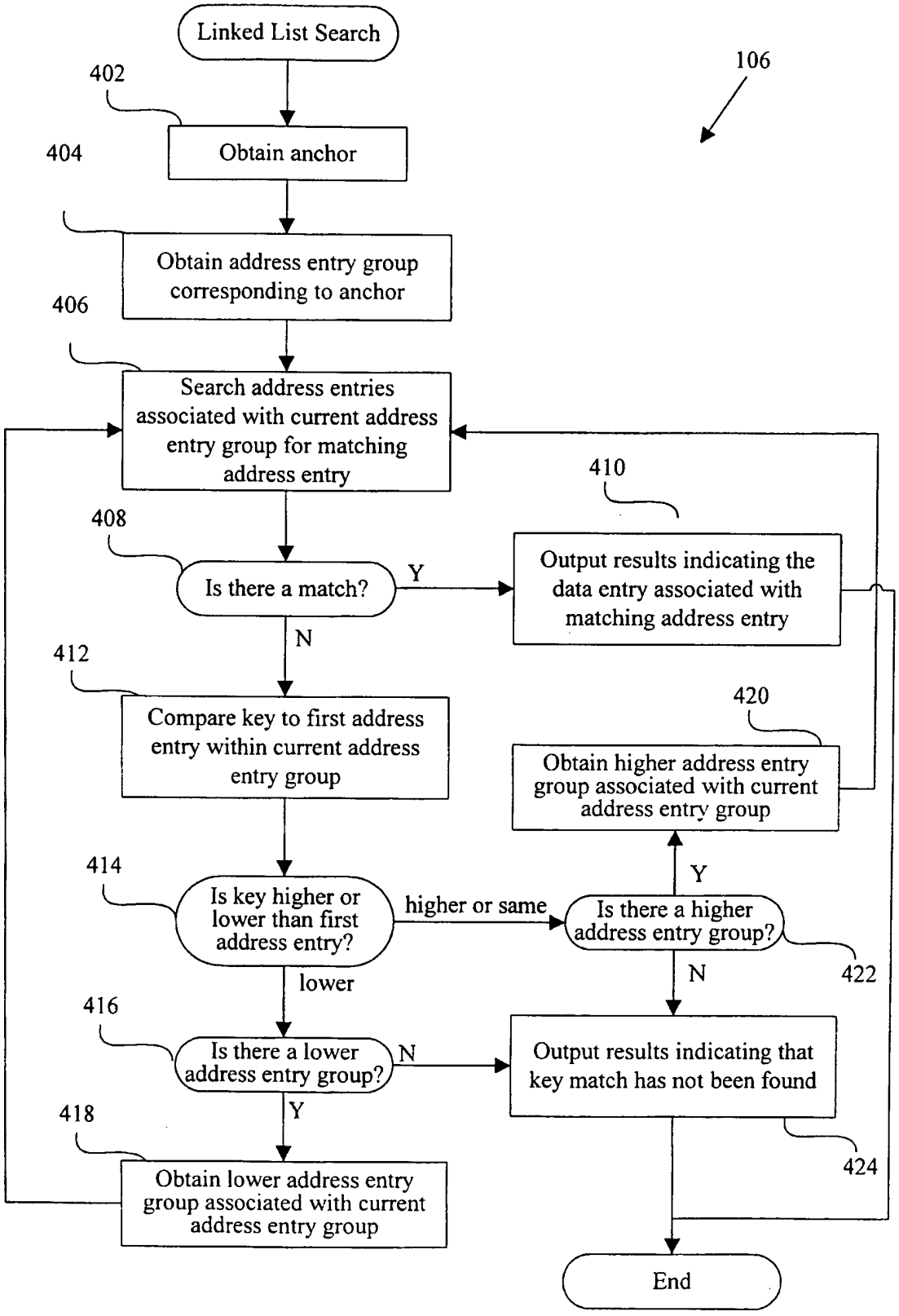

Linked List Search
106
402
Obtain anchor
404
Obtain address entry group corresponding to anchor
406
Search address entries associated with current address entry group for matching address entry
410
408
Is there a match?
Y
Output results indicating the data entry associated with matching address entry
N
412
Compare key to first address entry within current address entry group
420
Obtain higher address entry group associated with current address entry group
414
Is key higher or lower than first address entry?
higher or same
Is there a higher address entry group?
Y
422
N
lower
416
Is there a lower address entry group?
N
Output results indicating that key match has not been found
Y
418
424
Obtain lower address entry group associated with current address entry group
End

Figure 4

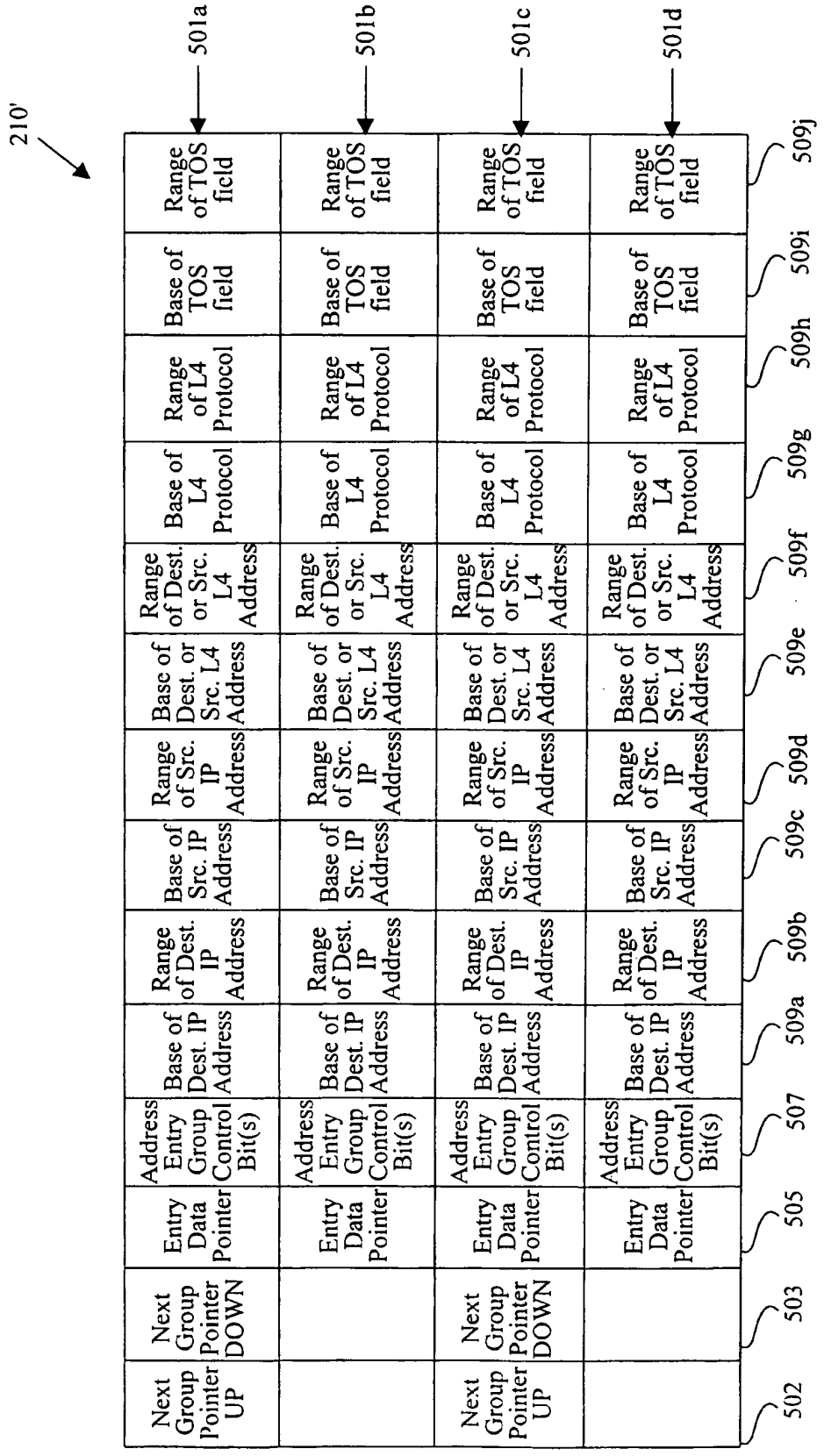

210'
501a
501b
501c
501d
Next Group Pointer UP
Next Group Pointer DOWN
Entry Data Pointer
Address Entry Group Control Bit(s)
Base of Dest. IP Address
Range of Dest. IP Address
Base of Src. IP Address
Range of Src. IP Address
Base of Dest. or Src. L4 Address
Range of Dest. or Src. L4 Address
Base of L4 Protocol
Range of L4 Protocol
Base of TOS field
Range of TOS field
502
503
505
507
509a
509b
509c
509d
509e
509f
509g
509h
509i
509j

Figure 5

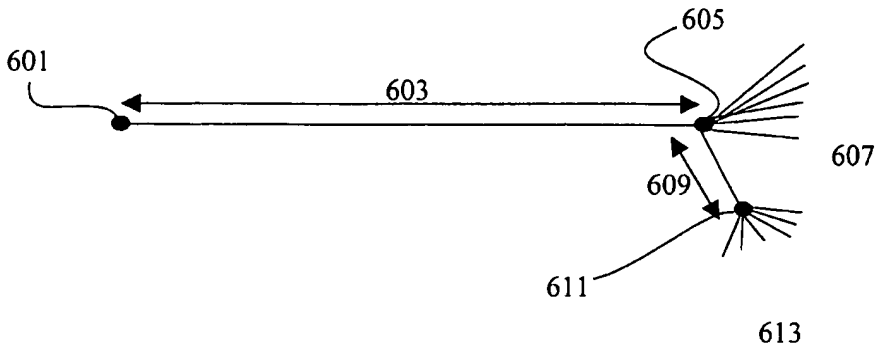


Figure 6

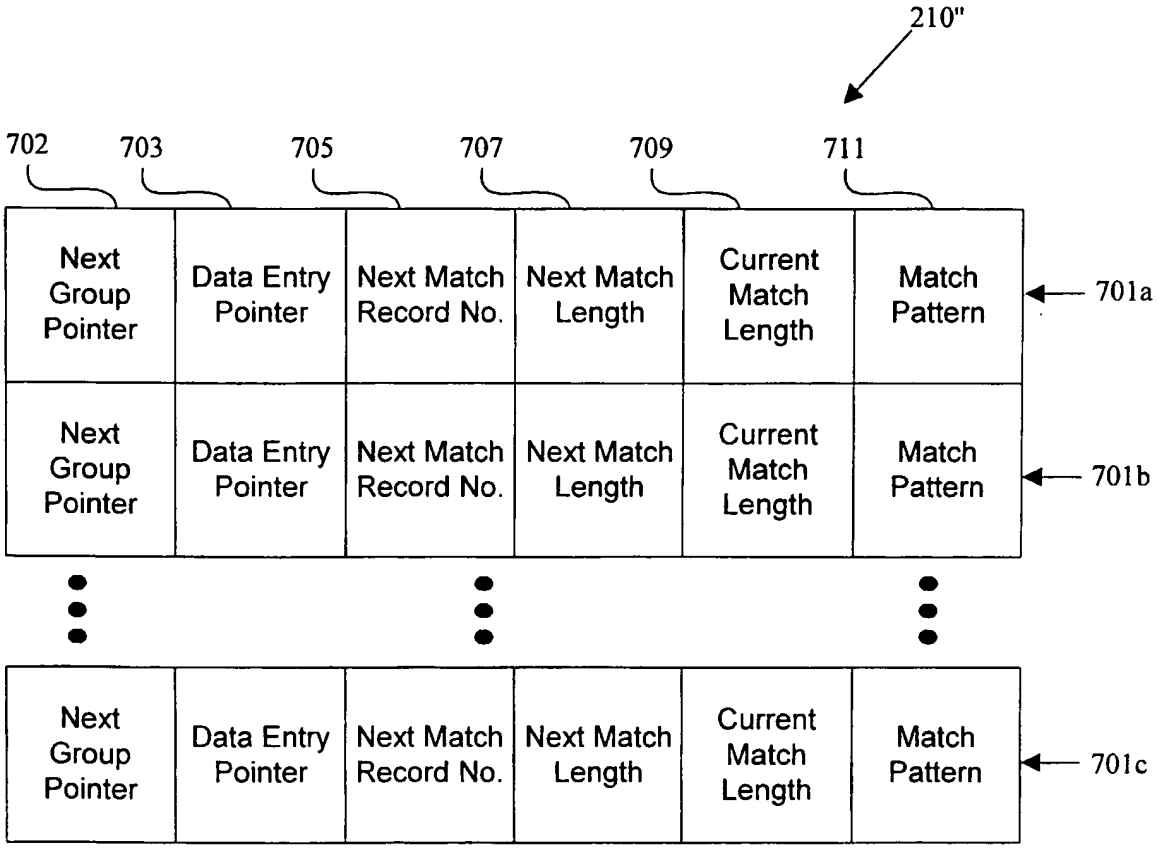


Figure 7

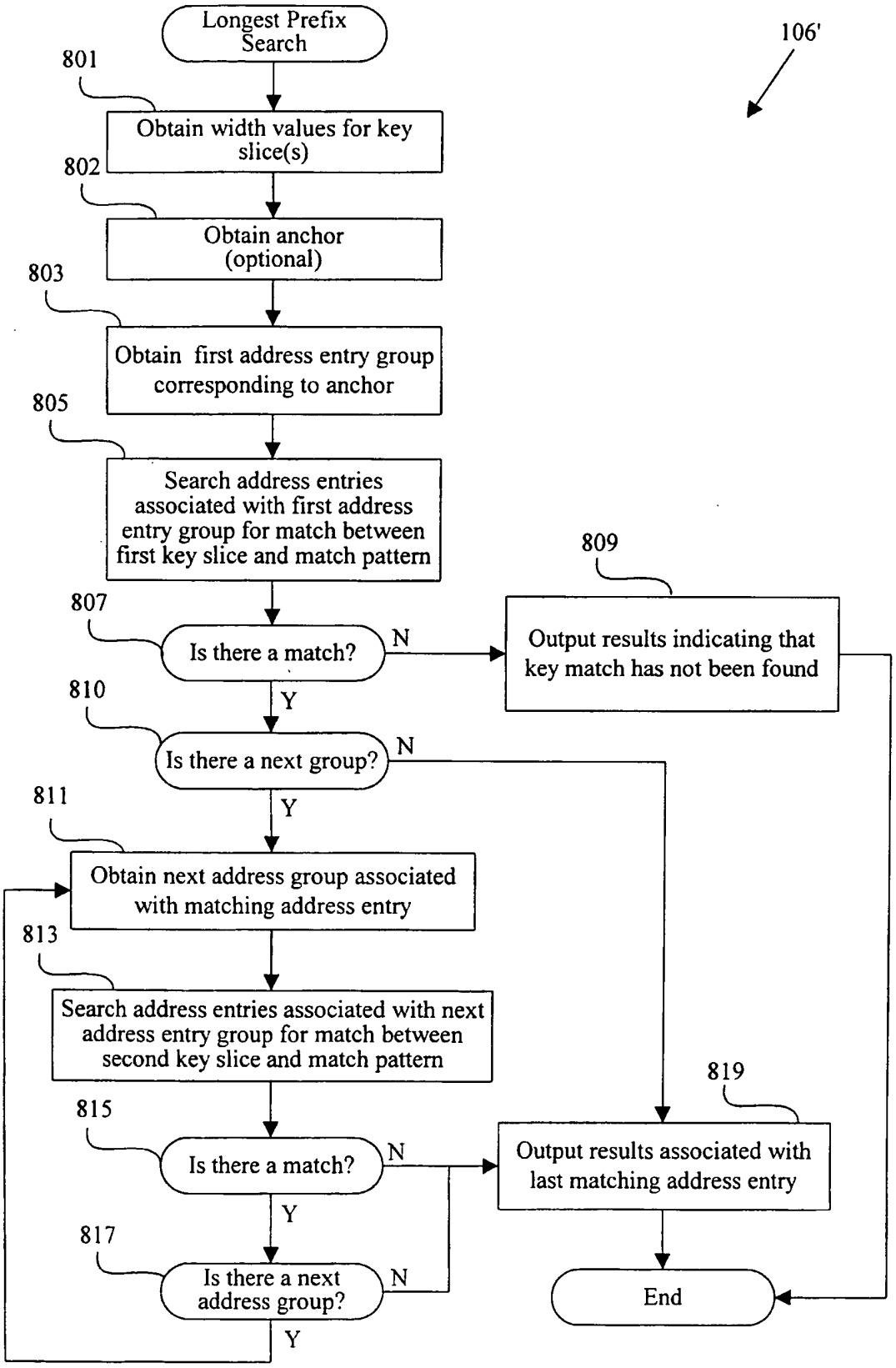

Longest Prefix Search
106'
801
Obtain width values for key slice(s)
802
Obtain anchor (optional)
803
Obtain first address entry group corresponding to anchor
805
Search address entries associated with first address entry group for match between first key slice and match pattern
807
Is there a match?
N
809
Output results indicating that key match has not been found
Y
810
Is there a next group?
N
Y
811
Obtain next address group associated with matching address entry
813
Search address entries associated with next address entry group for match between second key slice and match pattern
819
815
Is there a match?
N
Output results associated with last matching address entry
Y
817
Is there a next address group?
N
End
Y

Figure 8

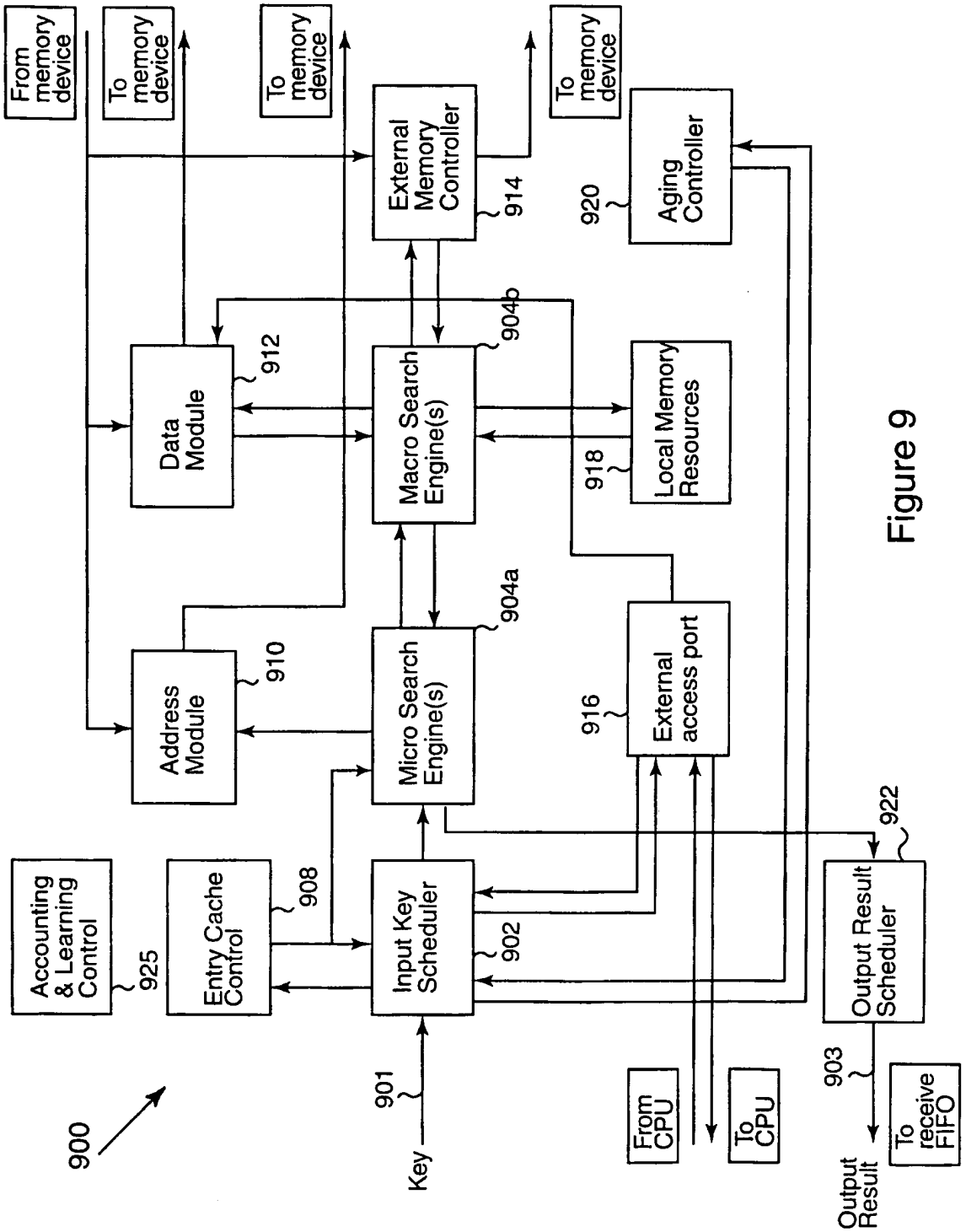

From memory device
To memory device
To memory device
To memory device
External Memory Controller
914
Aging Controller
920
Data Module
912
Macro Search Engine(s)
904b
Local Memory Resources
918
Address Module
910
Micro Search Engine(s)
904a
External access port
916
Accounting & Learning Control
925
Entry Cache Control
908
Input Key Scheduler
902
Output Result Scheduler
922
900
901
Key
From CPU
To CPU
903
Output Result
To receive FIFO

Figure 9

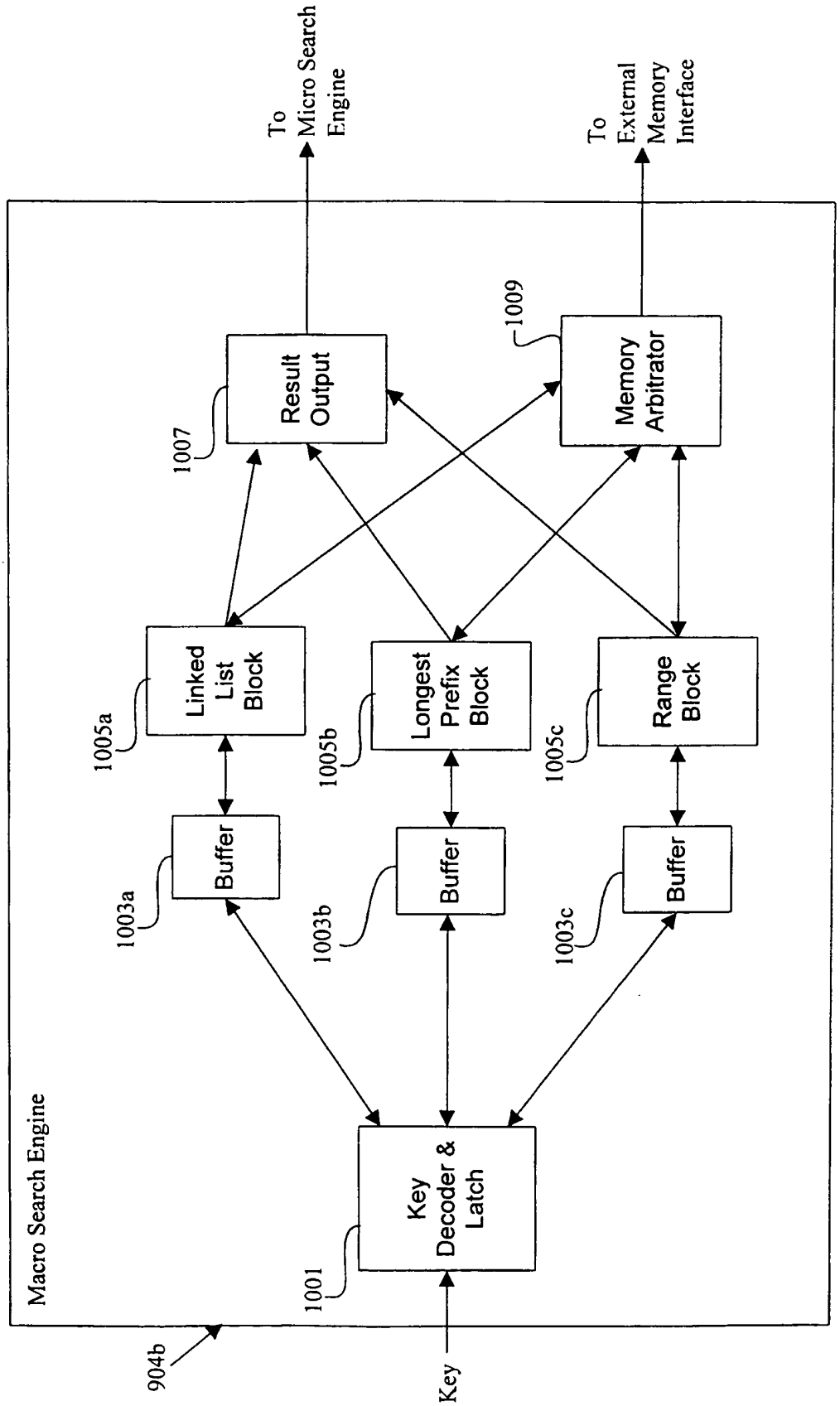

To Micro Search Engine
To External Memory Interface
Result Output
1007
Memory Arbitrator
1009
Linked List Block
1005a
Longest Prefix Block
1005b
Range Block
1005c
Buffer
1003a
Buffer
1003b
Buffer
1003c
Key Decoder & Latch
1001
Macro Search Engine
904b
Key

Figure 10

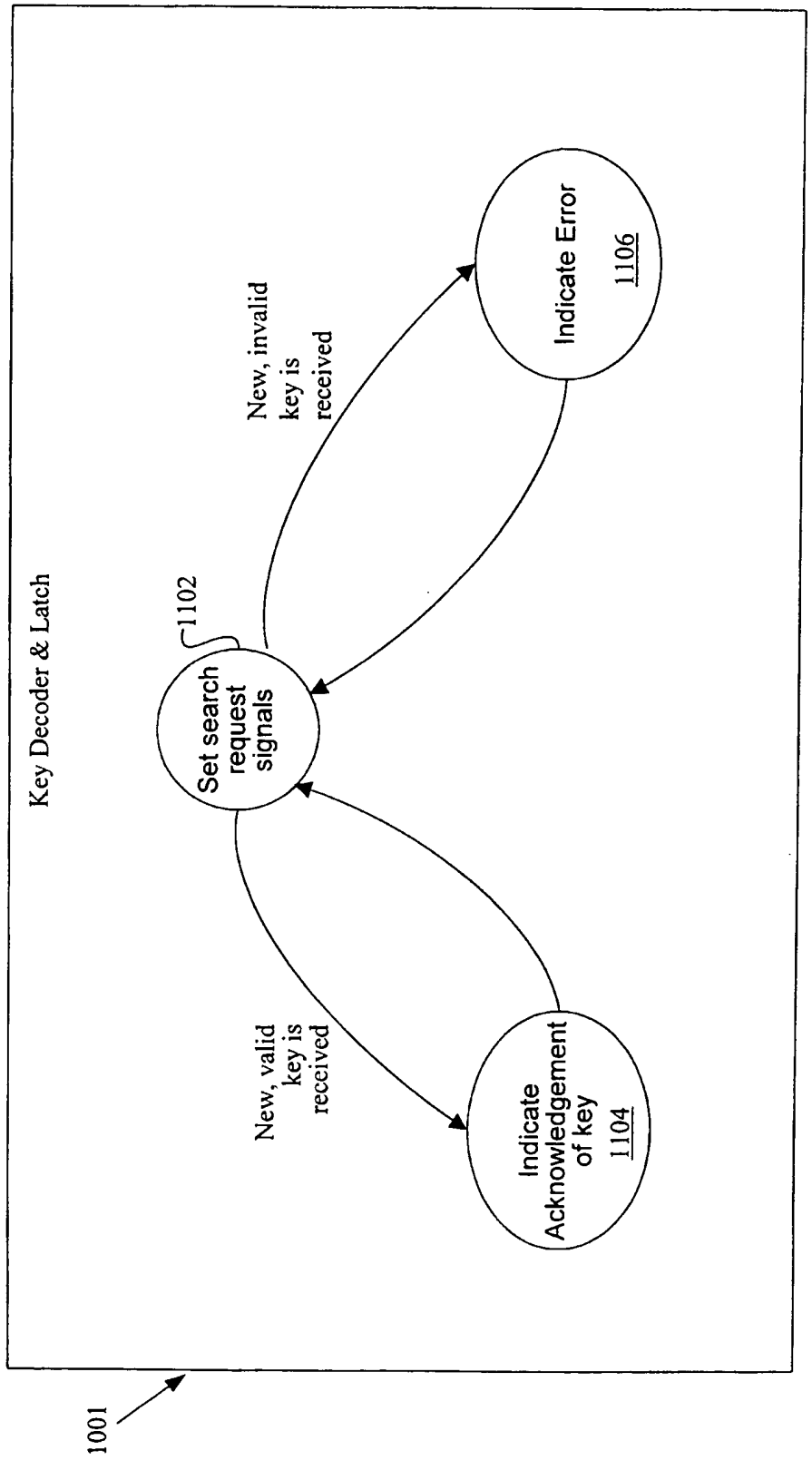

1001
Key Decoder & Latch
1102
Set search request signals
New, invalid key is received
Indicate Error 1106
New, valid key is received
Indicate Acknowledgement of key 1104

Figure 11

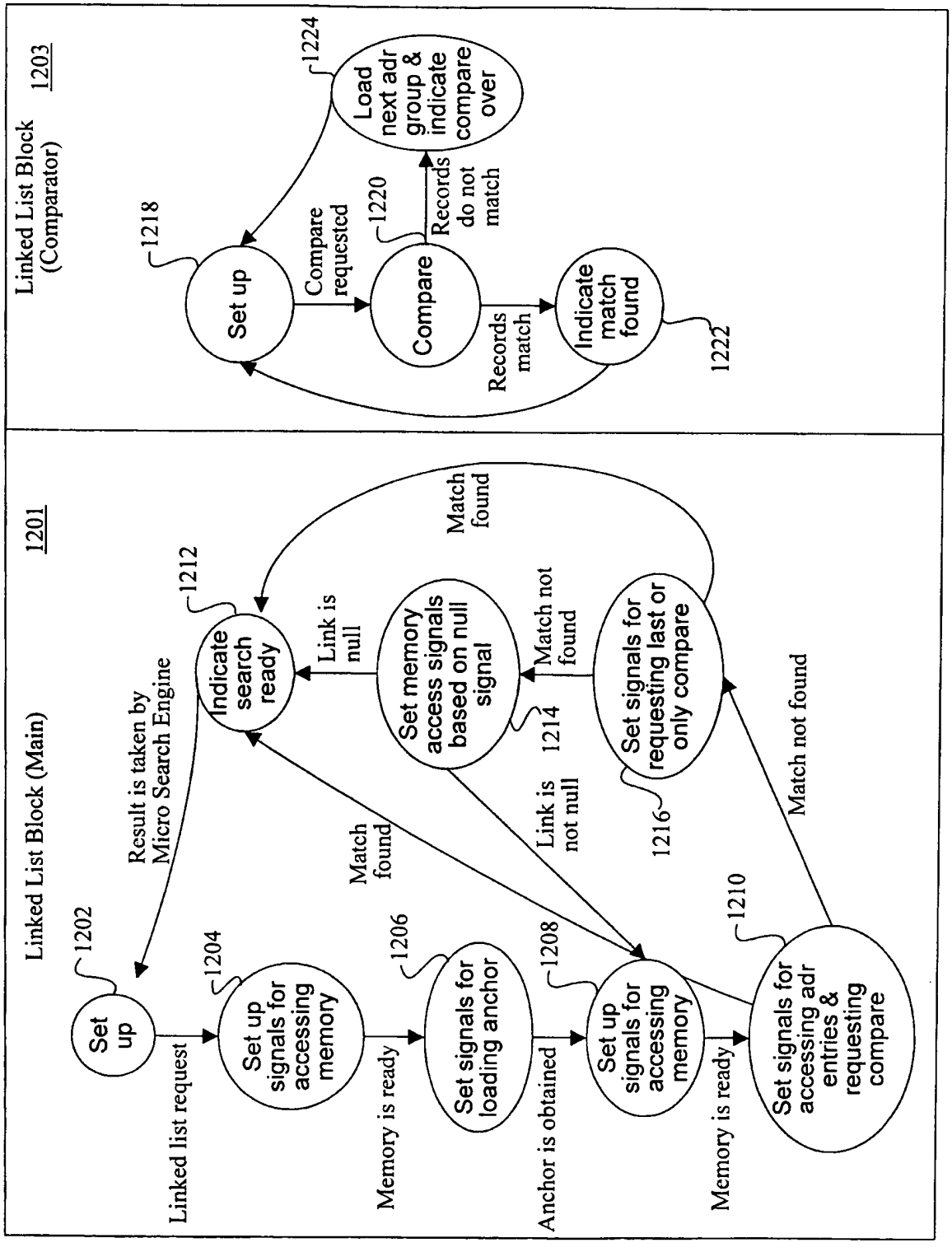

1005a
Linked List Block (Main) 1201
Linked List Block (Comparator) 1203
Set up
1202
Linked list request
Set up signals for accessing memory
1204
Memory is ready
Set signals for loading anchor
1206
Anchor is obtained
Set up signals for accessing memory
1208
Memory is ready
Set signals for accessing adr entries & requesting compare
1210
Match not found
Set signals for requesting last or only compare
1216
Match not found
Link is not null
Set memory access signals based on null signal
1214
Match found
Link is null
Indicate search ready
1212
Match found
Result is taken by Micro Search Engine
Set up
1218
Compare requested
Compare
1220
Records do not match
Load next adr group & indicate compare over
1224
Records match
Indicate match found
1222

Figure 12

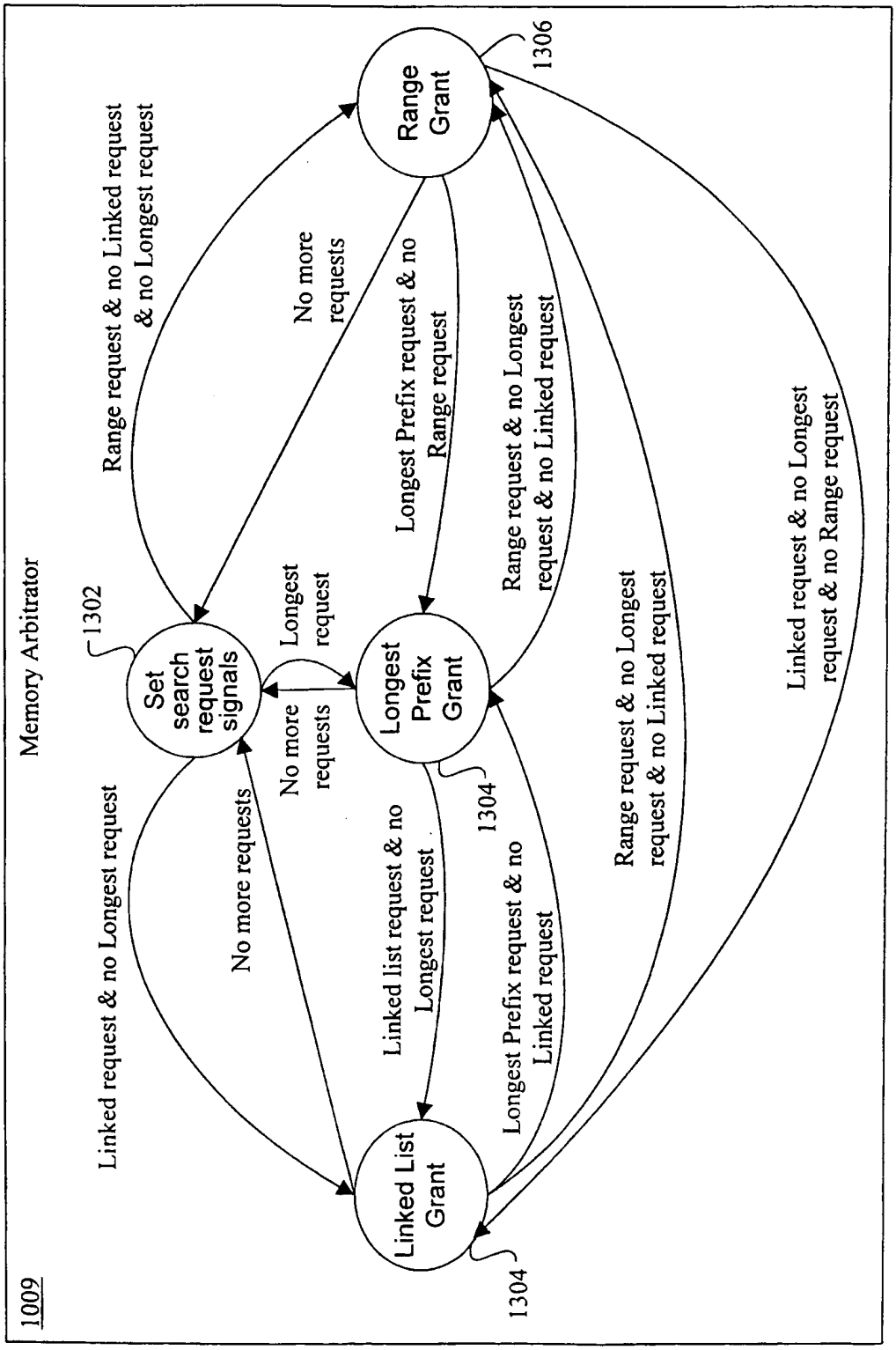

1009
Memory Arbitrator
Set search request signals
1302
Longest Prefix Grant
1304
Linked List Grant
1304
Range Grant
1306
Range request & no Linked request & no Longest request
No more requests
Longest request
No more requests
Linked request & no Longest request
No more requests
Linked list request & no Longest request
Longest Prefix request & no Linked request
Longest Prefix request & no Range request
Range request & no Longest request & no Linked request
Range request & no Longest request & no Linked request
Linked request & no Longest request & no Range request

Figure 13

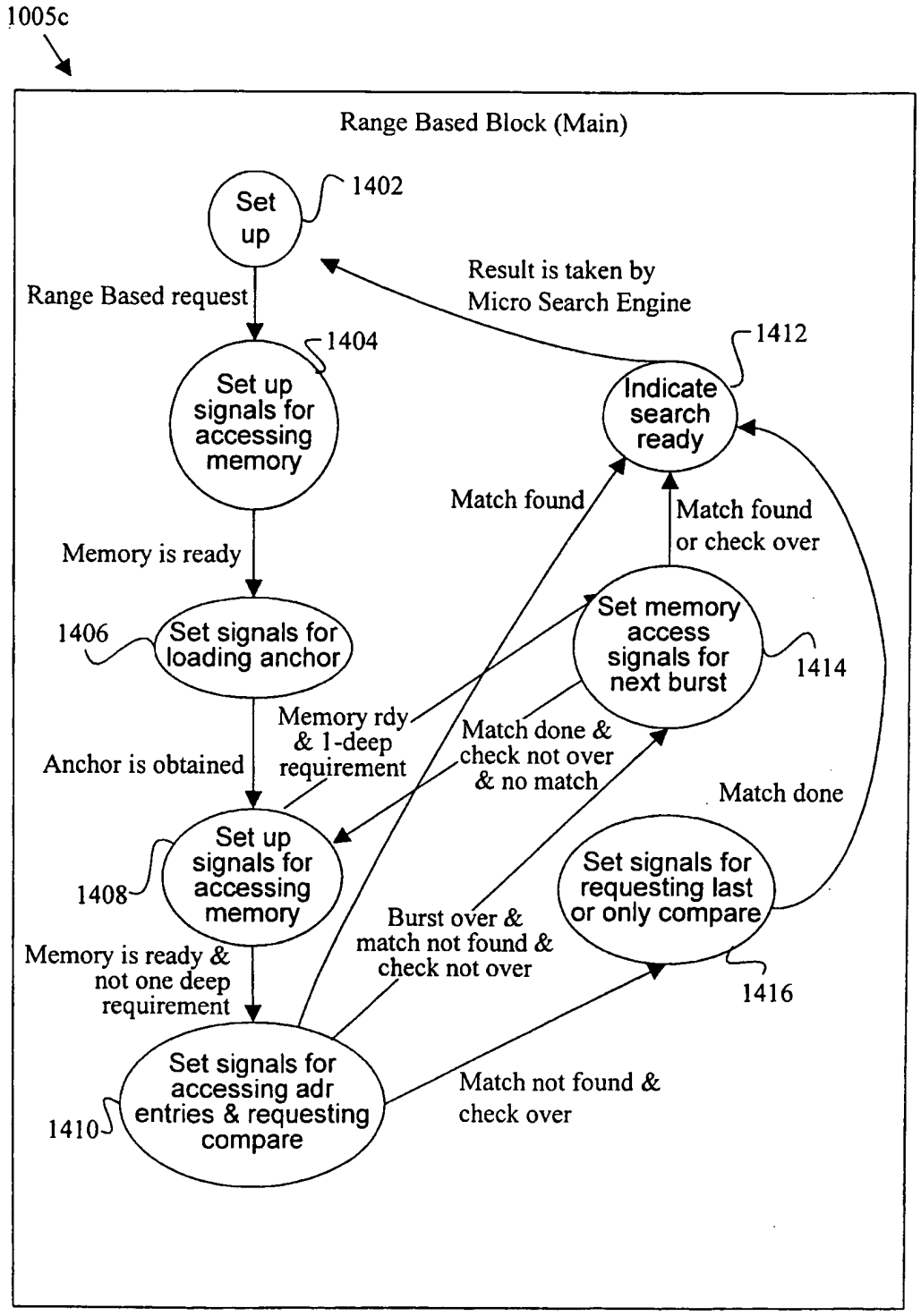

1005c
Range Based Block (Main)
Set up
1402
Range Based request
Result is taken by Micro Search Engine
1404
Set up signals for accessing memory
1412
Indicate search ready
Match found
Match found or check over
Memory is ready
1406
Set signals for loading anchor
Set memory access signals for next burst
1414
Memory rdy & 1-deep requirement
Match done & check not over & no match
Anchor is obtained
Match done
Set up signals for accessing memory
1408
Set signals for requesting last or only compare
Burst over & match not found & check not over
Memory is ready & not one deep requirement
1416
Set signals for accessing adr entries & requesting compare
Match not found & check over
1410

Figure 14A

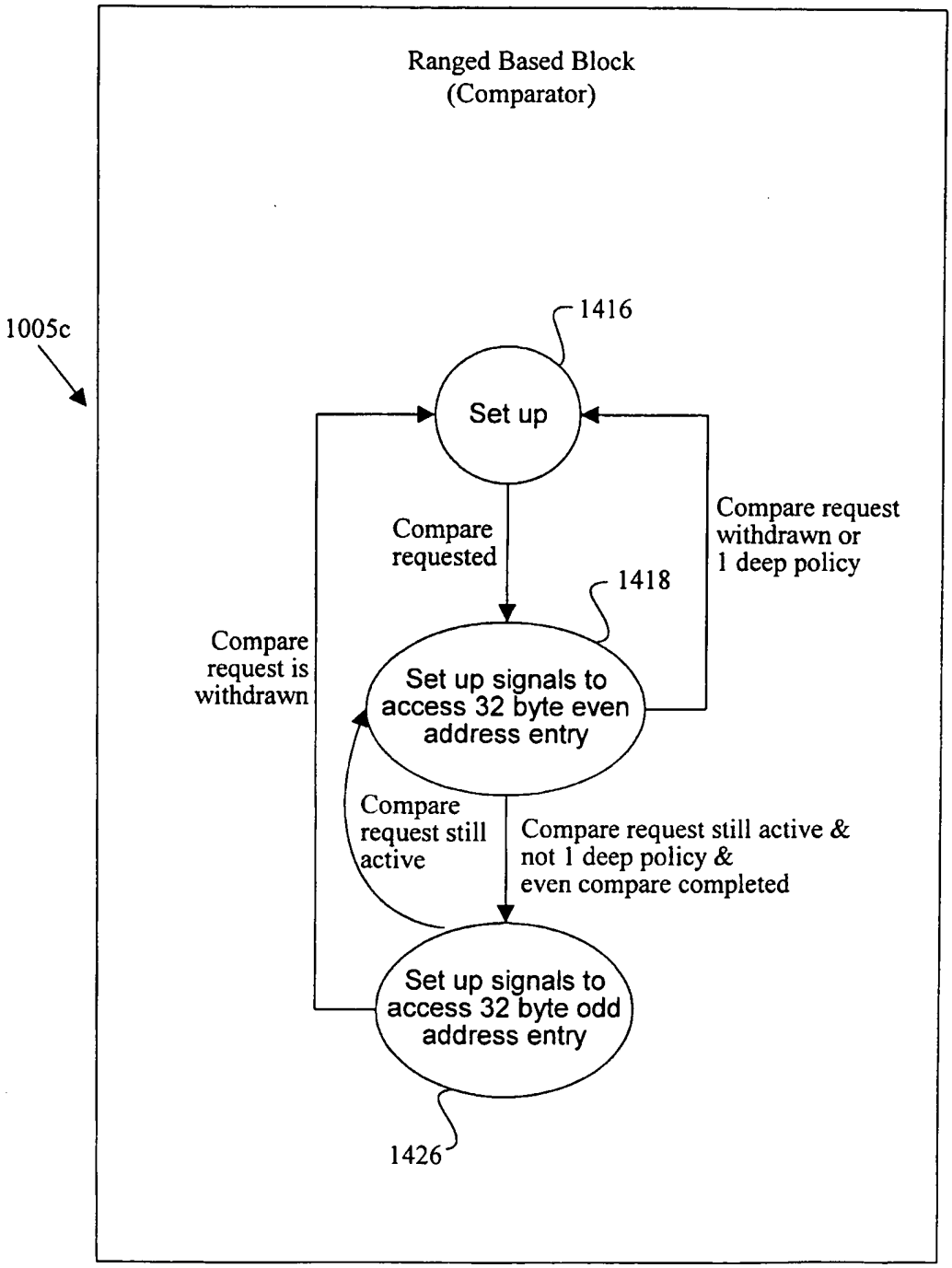

Ranged Based Block
(Comparator)
1005c
1416
Set up
Compare
requested
Compare request
withdrawn or
1 deep policy
1418
Compare
request is
withdrawn
Set up signals to
access 32 byte even
address entry
Compare
request still
active
Compare request still active &
not 1 deep policy &
even compare completed
Set up signals to
access 32 byte odd
address entry
1426

Figure 14B

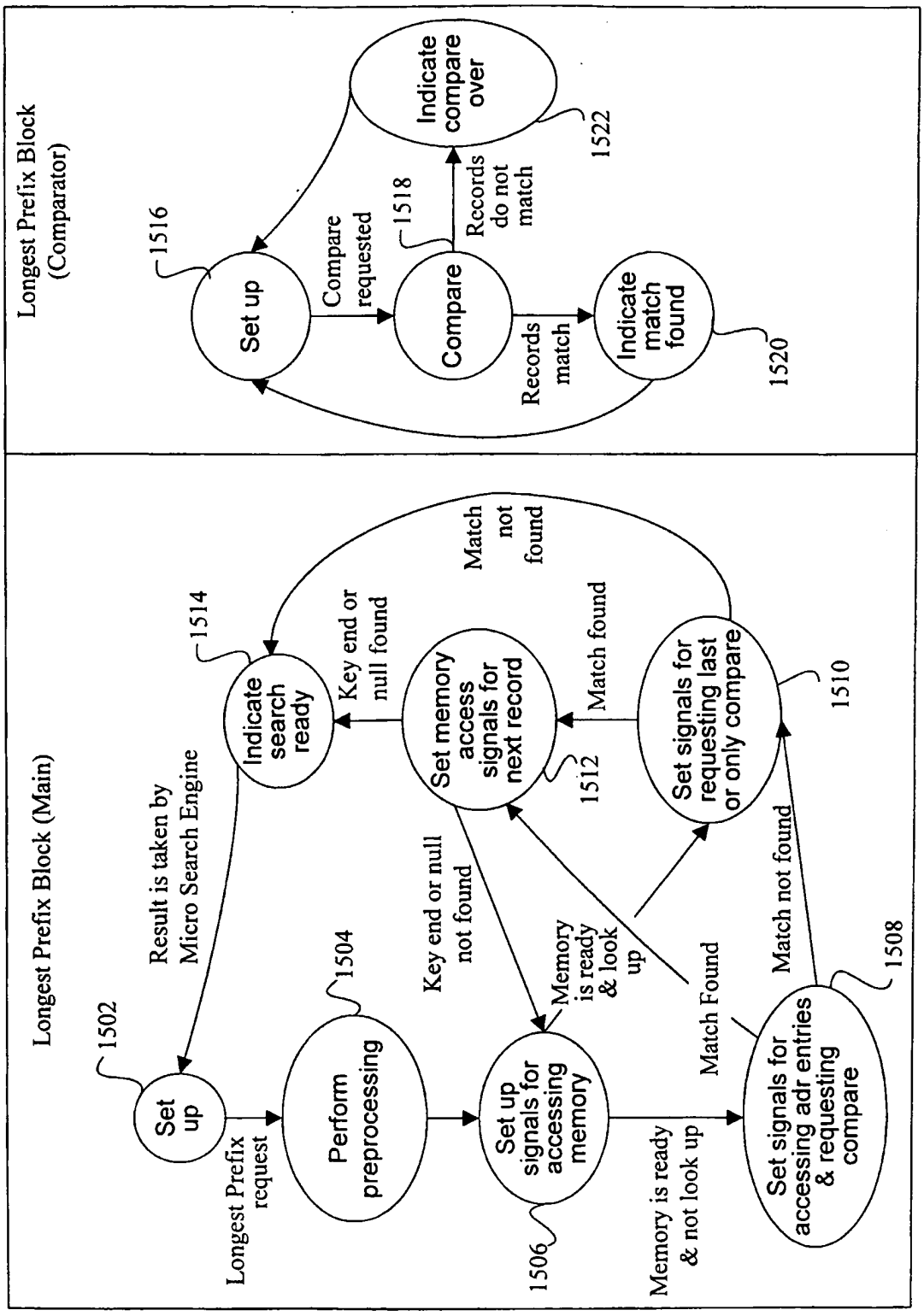

1005b
Longest Prefix Block (Main)
Longest Prefix Block (Comparator)
Set up
1502
Longest Prefix request
Perform preprocessing
1504
Result is taken by Micro Search Engine
Indicate search ready
1514
Key end or null found
Set memory access signals for next record
1512
Match found
Set signals for requesting last or only compare
1510
Match not found
Key end or null not found
Memory is ready & look up
Set up signals for accessing memory
1506
Memory is ready & not look up
Match Found
Set signals for accessing adr entries & requesting compare
1508
Match not found
Set up
1516
Compare requested
Compare
1518
Records do not match
Indicate compare over
1522
Records match
Indicate match found
1520

Figure 15

INTERNATIONAL SEARCH REPORT

Intern ial Application No
PCT/US 00/07416

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06F17/30

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
WPI Data, EPO-Internal, INSPEC, IBM-TDB

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 5 379 420 A (ULLNER MICHAEL K) 3 January 1995 (1995-01-03) column 2, line 19 -column 3, line 62 column 8, line 1 -column 9, line 51; figure 1A	1,21
X	US 5 909 686 A (HEJZA LEO ET AL) 1 June 1999 (1999-06-01) column 17, line 31 -column 18, line 21	1,21
X	US 5 950 191 A (SCHWARTZ ANDREW) 7 September 1999 (1999-09-07)	24
A	column 2, line 51 -column 3, line 2	1,25
A	US 5 787 430 A (NASSEHI MEHDI ET AL) 28 July 1998 (1998-07-28) column 4, line 1 -column 5, line 18	36,43

☐ Further documents are listed in the continuation of box C.

☒ Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
4 August 2000	11/08/2000

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Deane, E

1

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

information on patent family members

Inter nal Application No
PCT/US 00/07416

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 5379420 A	03-01-1995	NONE	
US 5909686 A	01-06-1999	EP 1010104 A	21-06-2000
		WO 9900750 A	07-01-1999
US 5950191 A	07-09-1999	NONE	
US 5787430 A	28-07-1998	WO 9600945 A	11-01-1996
		DE 69422935 D	09-03-2000
		EP 0804769 A	05-11-1997